OSTRAVSKÁ UNIVERZITA V OSTRAVĚ
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATIKY

TECHNIKA FUZZY MODELOVÁNÍ V ALGORITMICKÉM A PROGRAMOVÉM ZPRACOVÁNÍ

DOKTORSKÁ DISERTAČNÍ PRÁCE

AUTOR: Mgr. Viktor Pavliska
VEDOUCÍ PRÁCE: prof. Ing. Vilém Novák, DrSc.

2009

Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

Ostrava ....

(podpis)

Beru na vědomí, že tato doktorská disertační práce je majetkem Ostravské univerzity (autorský zákon Č. 121/2000 Sb., §60 odst. 1), bez jejího souhlasu nesmí být nic z obsahu práce publikováno.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Ostravské univerzity.

Ostrava ....

(podpis)

„Poslední krok rozumu je poznání,
že existuje nekonečně mnoho věcí,
které přesahují rozum.“

B. Pascal

Poděkování

Na tomto místě bych rád poděkoval především svému školiteli prof. Vilému Novákovi, kterému jsem vděčný hned v několika ohledech. Jednak mu děkuji za odborné rady související s prací samotnou, které ji pomáhaly postupně formovat až do současné podoby. Dále jsem mu vděčný, jako řediteli Ústavu pro výzkum a aplikace fuzzy modelování, za vytvoření těch nejlepších pracovních podmínek, jaké si jen lze představit. V neposlední řadě mu děkuji za bezedný zdroj povzbuzení, ze kterého jsem mnohdy čerpal energii pro práci. Bez nadsázky mohu říci, že bez něj by tato práce nemohla vůbec vzniknout.

Dále bych rád poděkoval prof. Irině Perfilievě, Martinu Štěpničkovi, Františku Huňkovi a Jaroslavu Knyblovi za jejich spolupráci a zdroj inspirace ze společných konzultací. Za programátorskou spolupráci na vyvíjených aplikacích děkuji Hashimu Habiballovi, Antonínu Dvořákovi a Marku Vajglovi. Zbývající části kolektivu děkuji za vytváření přátelské atmosféry, která panuje na pracovišti.

Nakonec musím vyjádřit díky své manželce za užitečné připomínky k jazykové úpravě práce a především ocenění za trpělivost, kterou projevila, když jsem se jí nemohl kvůli práci plně věnovat. Pevně doufám, že jí budu moci toto dočasné strádání v budoucnu vynahradit.

Abstract

Fuzzy logic initiated by L. A. Zadeh has already managed to make its way in various areas and branches, which is proving to last in the future, too. In order to implement it, supportive tools are needed, among others, to carry out its applications. This is also a long-term goal which is targeted by this work – to contribute to a gradually developing universal tool for support of fuzzy modelling.

The work is drawn in the way that we gradually learn both the theoretical bases and practical properties of algorithms leading to forming of basic building stones of the universal tool for fuzzy modelling. Hand in hand with the above-mentioned supporting elements, a partial base is laid for modelling of their mutual interconnections and combinations leading to creating of hybrid and hierarchical models of complex systems.

Chapter 2 is mainly devoted to fuzzy transform, which has already found its use in a lot of spheres. It is mentioned on the background of other fuzzy approximation techniques, which are often used in fuzzy modelling for approximation of real functions of one and more variables. Apart from the basic one-dimensional fuzzy transform, there are also generalized multi-dimensional variations in a form of Cartesian and triangular fuzzy transform. The question of fuzzy transform is also dealt with in more details from the point of view of algorithms, while the overwhelming majority is devoted to an analysis and estimation of computational complexity of the algorithms related to it.

Chapter 3 brings closer, apart from the perception-based logical deduction, which is our preferred tool for fuzzy modelling, a tool created on the basis of Petri nets – fuzzy Petri nets. These can serve for visualization of fuzzy IF–THEN rules, but above all they can be the roofing tool allowing their structuring in more complex systems, if interconnected to other tools and supporting algorithms of fuzzy modelling.

Chapter 4 describes basic types of fuzzy controllers and compares their parameters with their corresponding counterparts from the area of the classic theory of control. The end of the chapter introduces the simplest type of adaptive fuzzy controller constructed from own efficient controller which is supervised by a fuzzy controller that sets its parameters by an appropriate change of contexts of its linguistic variables.

Chapter 5 is devoted to the description of a performed practical application of a fuzzy controller for a transformation of pressure power transmission between two liquids, which was carried out within the programme of project consortia in the project FD-K3/033.

The closing chapter summarizes the results presented in this work and foreshadows its continuation in the future.

Keywords: *fuzzy modelling, fuzzy approximation, fuzzy control, signal filtering, Petri nets, rule based systems.*

Anotace

Fuzzy logika iniciovaná L. A. Zadehem se dnes již dokázala prosadit v mnoha oblastech a odvětvích, přičemž vše nasvědčuje tomu, že stejný trend bude pokračovat i do budoucna. Pro její uplatnění jsou, mimo jiné, zapotřebí podpůrné nástroje na realizaci jejích aplikací. To je také dlouhodobým cílem, který sleduje tato práce – přispět svým dílem do postupně se vyvíjejícího univerzálního nástroje pro podporu fuzzy modelování.

Práce je koncipována tak, že v jejím průběhu jsou postupně zaváděny jak teoretické základy, tak rovněž praktické náležitosti v podobě algoritmů, vedoucí k vytvoření základních stavebních kamenů univerzálního nástroje pro fuzzy modelování. Společně s těmito podpůrnými prvky je také zčásti položen základ pro modelování jejich vzájemného propojení a kombinace vedoucí k sestavení hybridních a hierarchických modelů komplexních systémů.

Kapitola 2 se věnuje především fuzzy transformaci, která již našla své uplatnění v mnoha oborech. Ta je uvedena na pozadí ostatních fuzzy aproximačních technik, které jsou často používány ve fuzzy modelování pro aproximaci reálných funkcí jedné i více proměnných. Vedle základní jednorozměrné fuzzy transformace jsou uvedeny také její zobecněné vícerozměrné varianty v podobě kartézské a triangulární fuzzy transformace. Problematika fuzzy transformace je podrobněji rozpracována také z algoritmického hlediska, přičemž velká část je věnována analýze a odhadu výpočetní náročnosti algoritmů s ní souvisejících.

V kapitole 3 je vedle logické dedukce na základě percepce pozorování, která je pro nás preferovaným existujícím nástrojem pro fuzzy modelování, vytvořen nástroj založený na Petriho sítích – *fuzzy Petriho sítě*. Ty mohou sloužit jednak k vizualizaci fuzzy IF–THEN pravidel, ale především ve spojení s ostatními nástroji a podpůrnými algoritmy fuzzy modelování mohou být tím zastřešujícím prostředkem, který umožní jejich skládání do složitějších soustav.

Kapitola 4 popisuje základní typy fuzzy regulátorů a srovnává jejich parametry se svými odpovídajícími protějšky z oblasti klasické teorie regulace. Na konci kapitoly je uveden nejjednodušší typ adaptivního fuzzy regulátoru sestaveného z vlastního výkonného regulátoru, nad kterým dohlíží fuzzy regulátor nastavující jeho parametry příslušnou změnou kontextů jeho jazykových proměnných.

Kapitola 5 je věnována popisu uskutečněné praktické aplikace fuzzy regulátoru pro transformátor přenosu tlakové energie mezi dvěma kapalinami, která byla realizována v rámci programu projektového konsorcia v projektu FD-K3/033.

Závěrečná kapitola shrnuje výsledky uvedené v této práci a naznačuje její pokračování do budoucna.

Klíčová slova: *fuzzy modelování, fuzzy aproximace, fuzzy regulace, filtrace signálu, Petriho sítě, pravidlové systémy.*

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů

$\emptyset$ – prázdná množina, 17
$\tilde{A}$ – fuzzy množina, 5
$A(x)$ – funkce příslušnosti fuzzy množiny $\tilde{A}$, 5
– stupeň příslušnosti prvku x do fuzzy množiny $\tilde{A}$, 5
$\tilde{A} \subsetneq U$ – $\tilde{A}$ je fuzzy množina na U, 5
R – fuzzy relace, 9
$R \circ \tilde{A}$ – obraz fuzzy množiny $\tilde{A}$ ve fuzzy relaci R, 10
$\wedge$ – infimum, minimum, 9
$\vee$ – supremum, maximum, 9
t – t-norma, 9
$\xrightarrow{t}$ – operace rezidua vzhledem k t-normě t, 9
U, V – univerza hodnot, 5
$\mathcal{F}(U)$ – množina všech fuzzy množin na univerzu U, 6
$U \times V$ – kartézský součin množin U a V, 9

& – logická spojka konjunkce, 9
A – predikátový symbol, 9
⇒ – logická spojka implikace, 9
DNF – disjuntivní normální forma, 9
CNF – konjunktivní normální forma, 9

d – rozměr prostoru, dimenzionalita úlohy, 16
$[a; b]$ – uzavřený spojitý interval hodnot $a \,..\, b$, 5
I^d – d-rozměrný interval $I^d \subseteq \mathbb{R}^d$, 16
$\mathbb{R}^d$ – d-rozměrný reálný prostor, 17

e_t – odchylka od požadovaného stavu v čase t, 64
Δe_t – změna odchylky, 66
$\Delta^2 e_t$ – druhá diference odchylky, 66
δe_t – kumulativní součet odchylky, 66
u_t – akční zásah, 68
Δu_t – změna akčního zásahu, 68

f^A – funkce představující aproximaci funkce f, 7
$\mathcal{D}(f)$ – definiční obor funkce f, 16
$f^{(-1)}_{\mathcal{P},\mathbf{F}}$ – inverzní F-transformace vůči komponentám $\mathbf{F}$, 15
$\mathbf{F}_{\mathcal{P}}[f]$ – přímá fuzzy transformace funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$, 14

$\mathbf{F}_k$ – vektor k reálných hodnot (komponent) $[F_1, \ldots, F_k]$, 13

G – generativní gramatika, 46
$\mathscr{L}$ – formální jazyk, 45
$\mathscr{A}, \mathscr{B}$ – jazykové výrazy, 8
IF–THEN – pravidlo typu jestliže – pak, 7
$\mathcal{R}$ – jazykové fuzzy pravidlo, 8

$\mathcal{N}$ – Petriho síť, 50

$\mathcal{O}$ – neostrý horní odhad používaný při určováni výpočetní složitosti, 22
$\mathcal{T}(A)$ – výpočetní složitost algoritmu A, 22

α, σ – symboly pro označení roviny, 17

$\mathcal{T}$ – triangularizace oblasti, 17
$\triangle(A, B, C)$ – trojúhelník o vrcholech A, B, C, 17
$|AB|$ – úsečka s krajními body A, B, 17
Ω – obecná oblast daná částí prostoru $\Omega \subseteq \mathbb{R}^d$, 17
ω – podoblast oblasti Ω, 18

$w(t)$ – časová funkce signálu, 33

Kapitola 1

Úvod

V současné době jsme svědky mohutného rozmachu informačních technologií, který je z valné části umožněn neustále se zvyšující výkonností počítačů. Tento trend se řídí tzv. Moorovým zákonem, podle kterého se počet součástek (tranzistorů a diod) v integrovaném obvodu zdvojnásobí vždy za dva roky, což představuje neuvěřitelný exponenciální růst. Tento zákon platí již od roku 1965, kdy byl vysloven Gordonem Moorem, ředitelem výzkumu a vývoje ve společnosti Fairchild Semiconductor jako odhad, a to jen čtyři roky poté, co byl poprvé použit plošný integrovaný obvod. V době, kdy své tvrzení vyslovil, obsahoval běžný integrovaný obvod přibližně 30 součástek a špičkový obvod měl kolem 60 komponent. V současnosti je také používána aktualizovaná varianta predikující zdvojnásobení počtu součástek v obvodu každých 18 měsíců.

Moorův odhad – ačkoli nese označení zákon – není zákonem fyzikálním, ale pouze empirickým, který růst hustoty tranzistorů v čase formuluje na základě odpozorovaného trendu. Nemá univerzální platnost, a jako takový samozřejmě vyvolává otázku, jak dlouho ještě bude pravdivý. Odhady se různí, ale jisté je, že existují fyzikální hranice toho, kolik součástek se na čip vejde. Moore předpovídá, že jeho zákon má před sebou ještě 10 až 20 let platnosti, než bude dosaženo hranice dané fyzikálními limity.

Společně s nárůstem výkonnosti počítačů, klesá obdobným tempem i jejich cena a tudíž se zlepšuje jejich dostupnost širšímu okruhu lidí. Kdysi byly počítače využívány hrstkou lidí ze specializovaných výpočetních středisek, avšak dnes je situace naprosto odlišná. S počítači již běžně pracuje široká veřejnost a zacházet s nimi se učí již děti na základních školách. Vše tedy nasvědčuje tomu, že se jejich uplatnění do budoucna bude ještě více rozšiřovat. Díky tomu, že jsou počítače čím dál tím dostupnější, přicházejí s nimi do styku i lidé bez specializovaného vzdělání, pro které by bylo žádoucí, kdyby své požadavky mohli klást počítači v přirozeném jazyce, jako by komunikovali s rovnocenným partnerem.

S tím jak se technika zdokonaluje, posouvají se i nároky na požadované úlohy. Kdysi byli lidé počítačům vděční za urychlení dlouhotrvajících jinak ručně prováděných výpočetních operací, jejichž zpracování na počítačích dnes považují za samozřejmost. Některé úlohy však nelze vyřešit pouze zvýšením výkonnosti stroje, na kterém je realizován. Existují oblasti, na jejichž zvládnutí nestačí pouhá *hrubá síla*. Dosud je například nevyřešena úloha překladu z jednoho jazyka do druhého na úrovni lidského tlumočníka nebo řízení automobilu v hustém provozu. Informační exploze, v podobě internetu přesyceného množstvím internetových stránek, dala zase vzniknout úloze jak v tom nepřeberném množství informací nalézt relevantní odpověď na kladený dotaz.

Počítače se samozřejmě stále používají i kvůli jejich rychlosti a pořád existují úlohy, na něž nestačí ani ty nejvýkonnější stroje – např. simulace řízené jaderné reakce na subatomární úrovni. Avšak pro běžného uživatele je tento potenciál nevyužitelný, dokud mu společně s výkonem není poskytnut i nástroj na jeho ovládání. Jednoduše řečeno, nepotřebujeme pouze „rychlejší kalkulačku“, ale jakousi *přidanou hodnotu* v podobě sofistikovaných algoritmů. Jinými slovy, software musí držet krok s vývojem hardware.

Tyto nároky se snaží uspokojit obor umělé inteligence, jehož nedílnou součástí je i rozpoznání a interpretace přirozeného jazyka, bez kterého není spousta současných úloh uspokojivě řešitelných. Pro práci s přirozeným jazykem se ukazuje jako nezbytné akceptování neurčitosti v podobě vágnosti, jakožto neoddělitelné součásti lidského uvažování a vyjadřování. Člověk totiž při řešení praktických úloh v životě, jako je např. řízení automobilu, neprovádí své úvahy na základě přesně naměřených hodnot, ale spíše „počítá“ s vágními pojmy typu „daleko, blízko“ nebo „rychle, pomalu“, tedy výrazy přirozeného jazyka.

Pro modelování fenoménu vágnosti se již osvědčila fuzzy logika. Ta právě představuje vhodný nástroj pro modelování alespoň úzké části přirozeného jazyka související s kvantitativním ohodnocováním jevů.

1.1 Práce se znalostmi

Člověk má jedinečnou schopnost ve způsobu, jak zachází s informacemi. V podstatě každý den jsme konfrontováni se spoustou situací, ve kterých se musíme rozhodovat na základě svých zkušeností nastřádaných během celého života. Již od dětství se formuje tato naše *znalostní báze*, přičemž s každou novou zažitou zkušeností se do ní přidávají, modifikují a případně i odstraňují další znalosti, čímž dochází k její průběžné aktualizaci.

Pracujeme s nimi jaksi mimoděk, v podstatě automaticky, takže si ani neuvědomujeme, jak jsou tyto nabyté zkušenosti ukládány a dávány do souvislostí s ostatními. Přičemž právě způsob reprezentace těchto znalostí je klíčovým faktorem určujícím možnosti jejich dalšího zpracování. Konečným cílem totiž je využití těchto znalostí pro řešení výše uvedených typů úloh. Otázkou tedy je, jakým způsobem předat tyto nabyté zkušenosti počítači, aby mohl zastávat minimálně stejně kvalifikované činnosti jako člověk.

Pokud sdělujeme své znalosti a zkušenosti jinému člověku, používáme k tomu především výrazových prostředků přirozeného jazyka. Odtud také pochází motivace pro reprezentaci znalostí ve formě jazykových IF–THEN pravidel. Obecně je tento způsob velmi univerzálním nástrojem, kterého se také hojně využívá téměř ve všech oblastech aplikací fuzzy logiky. Pro tento specifický rámec jsou používána pravidla speciálního tvaru – tzv. *fuzzy IF–THEN pravidla*. Jejich úzce specifikované označení je odvozeno jednak z toho, že jazyková část je modelována pomocí fuzzy množin a také způsob jejich interpretace a mechanismus pro odvození závěru je založen na fuzzy logice.

1.2 Aplikace fuzzy logiky

Nejčastější aplikace fuzzy logiky se vyskytují v oblasti regulace a řízení technologických procesů pomocí fuzzy regulátorů. Nespornou výhodou fuzzy regulátorů je v prvé řadě relativní jednoduchost jejich návrhu, která je téměř nezávislá na složitosti regulované soustavy, což je předurčuje k využití v mnoha aplikacích. Na základě zkušeností lze konstatovat, že čím je systém složitější

a komplexnější, tím je výhodnější jej řídit za pomoci fuzzy přístupu. To je dáno především tím, že v takovýchto případech bývá často jediná znalost o chování systému vyjádřena pomocí empirických a heuristických zkušeností experta, který se systémem již nějakou dobu pracuje. Fuzzy regulátor je pak navržen podle pravidel zadaných expertem, takže svou činností pak napodobuje jeho chování.

Tomu, že je fuzzy regulace úspěšný nástroj na regulaci a řízení, nasvědčuje také fakt, že již existuje nesčetné množství uskutečněných aplikací od inteligentních praček a vysavačů přes automatické řízení metra nebo výtahů v mrakodrapech až k vesmírným programům, kdy byl fuzzy regulátor například použit pro řízení automatického ramene při opravě Hubbleova teleskopu. Jednou z posledních úspěšných aplikací fuzzy regulace je také, v samostatné kapitole 5 této práce popsaný, fuzzy regulátor použitý pro transformátor tlakové energie mezi dvěma různorodými kapalinami.

Nemá samozřejmě smysl snažit se použít fuzzy regulátory za každou cenu ve všech případech. Tam, kde se již osvědčily klasické PID regulátory, není důvod situaci měnit. Fuzzy regulátory jsou spíše vhodnou alternativou pro případy, kdy klasická teorie regulace již nemůže poskytnout uspokojivé řešení. Budoucnost proto vidíme spíše v kombinaci obou přístupů, kdy je komplexní systém rozložen na dílčí oblasti, které jsou podle svého charakteru zpracovány odpovídajícím způsobem, přičemž nad celým systémem dohlíží fuzzy expertní systém v roli supervizora.

1.3 Přehled výsledků a přínos práce

Přínos této práce můžeme charakterizovat podle dosažených výsledků, které lze shrnout v následujících bodech:

1. Definování triangulárního a zobecněného d-rozměrného simplexového fuzzy rozkladu pro variantu triangulární, resp. d-simplexové fuzzy transformace.

2. Návrh algoritmů fuzzy transformace společně s odhadem jejich strukturální výpočetní složitosti v závislosti na použitém fuzzy rozkladu a různých vlastnostech vstupních dat z hlediska jejich uspořádání.

3. Implementace algoritmů kartézské vícerozměrné fuzzy transformace a její začlenění do uživatelského prostředí systému LFLC 2000. Speciálně dvojrozměrná fuzzy transformace byla použita v aplikaci pro kompresi obrázků.

4. Implementace evolučních algoritmů na základě diferenciální evoluce a jejich uplatnění na optimalizaci rozložení uzlů fuzzy rozkladu pro minimalizaci aproximační chyby fuzzy transformace.

5. Návrh a implementace různých variant fuzzy filtrů a odhad jejich konkrétní výpočetní složitosti v podobě počtu elementárních aritmetických operací nutných k provedení výpočtu hodnoty výstupního filtrovaného signálu. Fuzzy filtr byl použit v následujících situacích:

   - Inverzní fuzzy filtr byl začleněn do systému LFLC 2000 na úpravu průběhu výsledné funkce inferenčního mechanismu pro libovolnou kombinaci inferenční a defuzifikační metody.
   - Varianta klouzavého fuzzy filtru byla použita při implementaci tzv. *hladké dedukce*, která je rovněž součástí softwarového balíku LFLC 2000.

- V rámci účasti v soutěži NN3 byla vytvořena aplikace používající inverzní fuzzy filtr pro analýzu a následnou predikci časových řad.

6. Návrh a implementace nástroje pro modelování a vizualizaci fuzzy IF–THEN pravidel s možností jeho dalšího rozšiřování do podoby univerzálního fuzzy modelovacího nástroje kombinujícího různé techniky pro tvorbu hybridních a hierarchických modelů.

7. Vytvoření přehledu nejčastěji používaných základních typů fuzzy regulátorů společně se srovnáním jejich parametrů s odpovídajícími variantami regulátorů z klasické teorie regulace.

8. Návrh a implementace adaptivního fuzzy regulátoru sestaveného jako spojení dohlížecího fuzzy regulátoru nastavujícího jazykové kontexty výkonného fuzzy regulátoru.

9. Realizace regulace transformátoru přenosu tlakové energie mezi dvěma kapalinami v rámci řešení projektu FD-K3/033.

Kapitola 2

Fuzzy aproximace

Hlavním cílem fuzzy aproximace, který budeme v dalším textu sledovat, je využití nástrojů fuzzy logiky pro aproximaci funkcí, přičemž nejčastěji používaným prostředkem jsou *fuzzy množiny*. Fuzzy množiny jsou funkce z nějaké množiny (univerza) do algebry pravdivostních hodnot. Algebra pravdivostních hodnot je zpravidla MTL-algebra, BL-algebra, MV-algebra aj., což jsou speciální případy tzv. reziduovaného svazu [19]. V této práci se zpravidla omezíme jen na standardní Łukasiewiczovu MV-algebru.

Definice 1 (Standardní Łukasiewiczova MV-algebra).
Standardní Łukasiewiczova MV-algebra je struktura

$$\mathscr{L} = \langle [0;1], \wedge, \vee, \otimes, \rightarrow, 0, 1 \rangle, \tag{2.1}$$

kde $\wedge$ je operace minima, $\vee$ je operace maxima, $\otimes, \rightarrow$ jsou Łukasiewiczova konjunkce a implikace definované takto:

$$a \otimes b = \max\{0, a + b - 1\}, \tag{2.2}$$

$$a \rightarrow b = \min\{1, 1 - a + b\} \tag{2.3}$$

pro všechna $a, b \in [0;1]$.

Poznámka 1.
Operace $\wedge, \otimes$ jsou speciálním případem tzv. t-norem [17, 19], což jsou binární operace t na $[0;1]$, které jsou asociativní, komutativní, monotónní vzhledem k $\leq$ a omezené, tj. $t(a,1) = a$, $a \in [0;1]$.

Definice 2 (Fuzzy množina).
Fuzzy množina $\tilde{A}$ na univerzu U, což značíme pomocí $\tilde{A} \subseteq U$, je zobecněním klasické množiny $A \subseteq U$ v tom smyslu, že její charakteristická funkce $A(x)$, $x \in U$, nazývaná též jako *funkce příslušnosti*, nabývá hodnot (*stupňů příslušnosti*) z nosiče uvažované algebry pravdivostních hodnot. V této práci budeme používat interval $[0;1]$, a proto definujeme fuzzy množinu jako funkci

$$A(x) : U \longrightarrow [0;1]$$

Definice 3 (Základna fuzzy množiny).
Mějme dánu fuzzy množinu $\tilde{A} \subseteq U$. Pomocí $\mathrm{Supp}(\tilde{A})$ značíme *základnu* (support) fuzzy množiny představující (klasickou) množinu prvků s nenulovým stupněm příslušnosti.

$$\mathrm{Supp}(\tilde{A}) = \{x \mid x \in U, A(x) > 0\}$$

Definice 4 (Normální fuzzy množina).
Fuzzy množina $\tilde{A} \subsetneq U$ je *normální*, pokud alespoň jeden její prvek má stupeň příslušnosti roven 1.

$$(\exists x \in U) : A(x) = 1$$

Poznámka 2.
V dalším textu budeme pracovat výhradně s fuzzy množinami nad číselnými (reálnými) univerzy. Pokud tedy nebude uvedeno jinak, budeme dále předpokládat, že univerza U fuzzy množin jsou tvořena prvky vektorového prostoru $\mathbb{R}^d$, $d \geq 1$.

Definice 5 (Konvexní fuzzy množina).
Nechť $U \subseteq \mathbb{R}^d$ je konvexní množina. Fuzzy množina $\tilde{A} \subsetneq U$ je *konvexní*, pokud:

$$(\forall x, y \in U)(\forall \lambda \in [0;1]) : A(\lambda x + (1-\lambda)y) \geq A(x) \wedge A(y).$$

Definice 6 (Fuzzy pokrytí množiny).
Nechť U je množina a $\mathcal{C} = \{\tilde{A}_i, \ldots, \tilde{A}_n\}$ je systém fuzzy množin. Říkáme, že tento systém tvoří *fuzzy pokrytí* množiny U, jestliže

$$U \subseteq \bigcup_{i=1}^{n} \mathrm{Supp}(\tilde{A}_i),$$

tedy, že každý prvek z množiny U patří do některé fuzzy množiny $\tilde{A}_i, \ldots, \tilde{A}_n$ s nenulovým stupněm příslušnosti.

Definice 7 (Množina všech fuzzy množin).
Nechť U je množina, potom pomocí $\mathcal{F}(U)$ značíme množinu všech fuzzy množin nad univerzem U.

$$\mathcal{F}(U) = \{A \mid A \subsetneq U\}$$

Definice 8 (Fuzzy funkce typu 1 [16]).
Nechť U, V jsou množiny. *Fuzzy funkcí typu 1* rozumíme funkci:

$$\tilde{f} : U \longrightarrow \mathcal{F}(V),$$

která přiřazuje prvkům z univerza U fuzzy množiny nad univerzem V.

Definice 9 (Fuzzy funkce typu 2 [16]).
Nechť U, V jsou množiny. *Fuzzy funkcí typu 2* rozumíme funkci:

$$\tilde{f} : \mathcal{F}(U) \longrightarrow \mathcal{F}(V),$$

což je v podstatě klasická funkce nad množinami fuzzy množin, která přiřazuje fuzzy množinám nad univerzem U fuzzy množiny nad univerzem V.

Poznámka 3.
Fuzzy funkce typu 2 se nejčastěji vyskytují v současných řídících systémech s implementovanou fuzzy logikou. Stav ovládaného systému bývá reprezentován vektorem číselným hodnot jednotlivých sledovaných veličin. Jednotlivé složky tohoto vektoru se nejprve fuzzifikují a následně použijí jako vstupní hodnoty do fuzzy funkce typu 2, představující vlastní výkonnou část fuzzy regulátoru, čímž je získána výstupní fuzzy množina, která se posléze defuzzifikuje vhodně zvolenou metodou na číselnou hodnotu představující konečný akční zásah do řízeného systému. Výše zmíněná fuzzifikace je typickým příkladem fuzzy funkce typu 1.

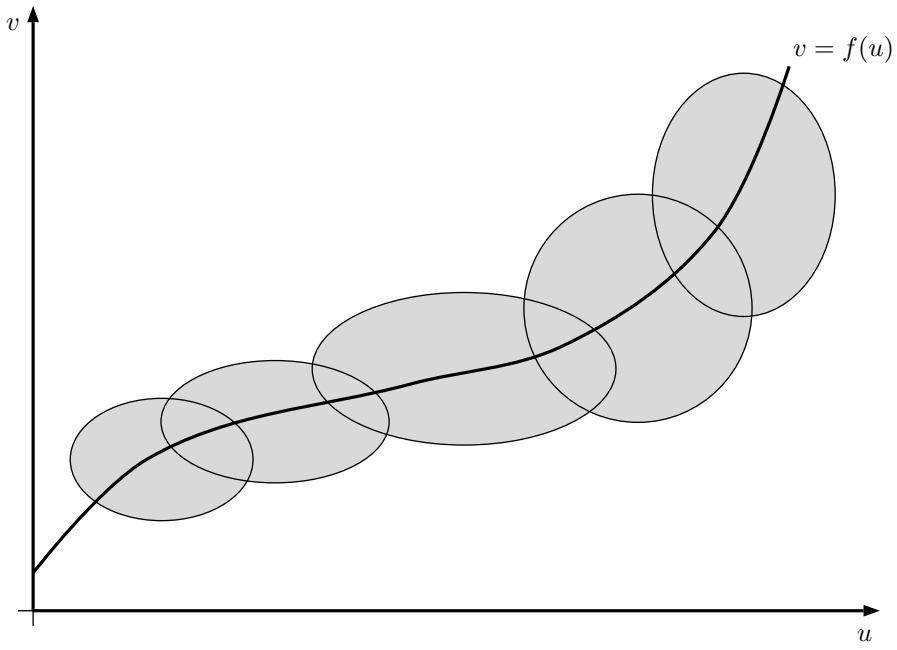
Stávající přístup k problematice fuzzy aproximace je možno shrnout do tří hlavních oblastí. Jedná se především o využití aproximačních vlastností následujících prostředků:

- Fuzzy IF–THEN pravidel
- Takagi-Sugeno pravidel
- Fuzzy transformace

V dalším textu následuje jejich stručný popis, přičemž hlavní část bude věnována fuzzy transformaci.

2.1 Fuzzy aproximace pomocí fuzzy IF–THEN pravidel

Výchozím bodem při použití fuzzy IF–THEN pravidel je situace, kdy máme pouze hrubou představu o průběhu funkce f a hledáme způsob, jak ji realizovat pomocí fuzzy relací. Předpokládejme, že hodnoty původní funkce f známe v několika nepřesně specifikovaných oblastech, které jsou na obr. 2.1 zobrazeny v podobě elips. Na základě této částečné informace o původní funkci se pak snažíme vytvořit funkci f^A jakožto její aproximaci. Pro jednoduchost se v dalším zaměříme pouze na funkce jedné vstupní proměnné.



Obrázek 2.1: Hrubá představa o průběhu funkce f.

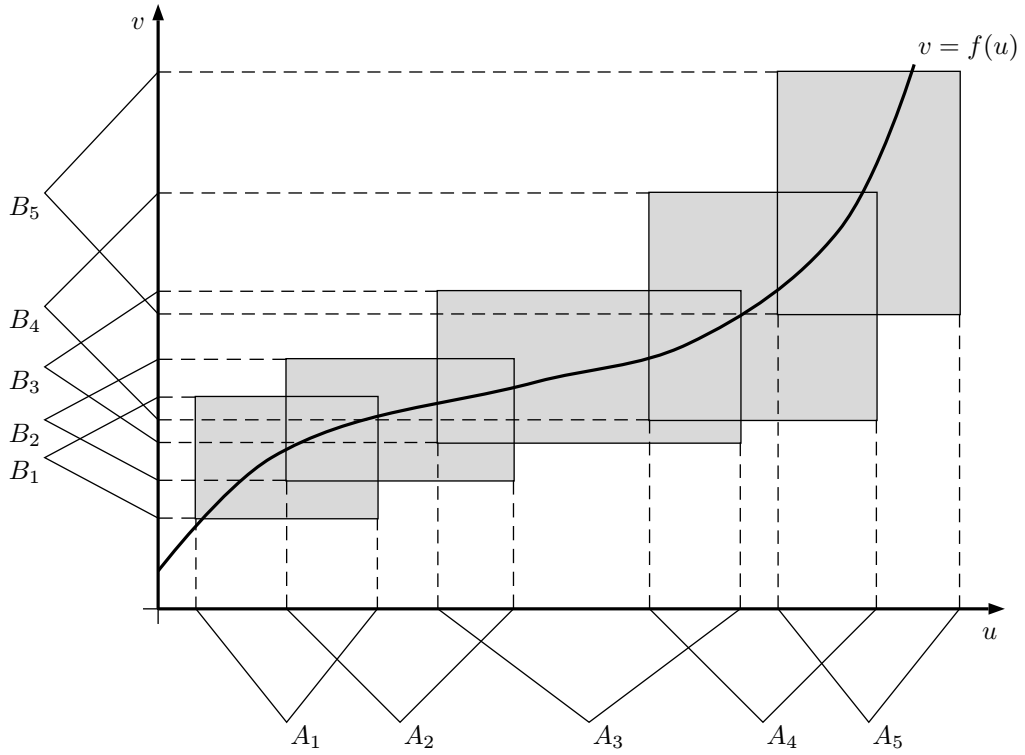
Je-li fuzzy funkce konečná[1], můžeme ji reprezentovat pomocí tabulky 2.1, kterou chápeme přirozeně tak, že vstupní hodnotě $\tilde{A}_i$ odpovídá výstupní hodnota $\tilde{B}_i$ pro všechna $1 \leq i \leq m$, přičemž $\tilde{A}_1, \ldots, \tilde{A}_m \underset{\sim}{\subseteq} U$ a $\tilde{B}_1, \ldots, \tilde{B}_m \underset{\sim}{\subseteq} V$ a je požadováno, aby fuzzy množiny $\tilde{A}_1, \ldots, \tilde{A}_m$ a popřípadě také $\tilde{B}_1, \ldots, \tilde{B}_m$ tvořily *fuzzy pokrytí* (viz definice 6) množiny U resp. V.

[1] Ve smyslu, že její definiční obor je konečná množina.

$$\tilde{f}: \begin{array}{c|c|c|c|c} x & \tilde{A}_1 & \tilde{A}_2 & \cdots & \tilde{A}_m \\ \hline y & \tilde{B}_1 & \tilde{B}_2 & \cdots & \tilde{B}_m \end{array}$$

Tabulka 2.1: Reprezentace konečné fuzzy funkce tabulkou.

Takováto reprezentace fuzzy funkce typu 2 pomocí dvojic $(\tilde{A}_i, \tilde{B}_i)$ se nazývá *fuzzy graf* [47]. Nejlépe lze danou situaci ilustrovat obrázkem 2.2.



Obrázek 2.2: Schématická ilustrace fuzzy funkce (grafu) $\tilde{f}$ aproximující funkci f. Pod osou u a vlevo od osy v jsou znázorněny funkce příslušnosti fuzzy množin $\tilde{A}_i$ a $\tilde{B}_i$

Nyní nás bude zajímat způsob, jak z daného fuzzy grafu dostaneme konkrétní funkci f. Vzhledem k tomu, že však o ní máme jen hrubou představu, nedostaneme ve skutečnosti tuto funkci, ale jen nějakou její aproximaci f^A. Pokud máme fuzzy graf zadán tabulkou ve tvaru 2.1, můžeme z ní přímo sestavit soustavu fuzzy IF–THEN pravidel ve tvaru:

$$\begin{array}{ll} \mathcal{R}_1: & \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}_1\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}_1 \\ \mathcal{R}_2: & \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}_2\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}_2 \\ \vdots & \vdots \quad \vdots \quad \vdots \quad \vdots \\ \mathcal{R}_m: & \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}_m\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}_m \end{array} \tag{2.4}$$

tvořící *jazykový popis*, kde každé $\mathscr{A}_i$, resp. $\mathscr{B}_i$ je tzv. *jazykový výraz* představující symbolický název příslušející k odpovídajícím fuzzy množinám $\tilde{A}_i$, resp. $\tilde{B}_i$. Každé pravidlo $\mathcal{R}_i$ takovéhoto

jazykového popisu (2.4) je potom lokálním popisem chování funkce $\tilde{f}$ na oblasti pokryté odpovídajícími fuzzy množinami. V dalším se zaměříme na interpretaci jazykového popisu (2.4).

Interpretace jazykového popisu probíhá ve dvou krocích. V prvním kroku je popisu přiřazena jedna ze dvou forem: *disjunktivní normální forma* nebo *konjunktivní normální forma*. Předpokládáme, že máme dán jazyk predikátové fuzzy logiky, který obsahuje spojky *konjunkce* ($\wedge$) a *silné konjunkce* ($\mathbf{\&}$), disjunkce ($\vee$) a implikace ($\Rightarrow$), popř. kvantifikátory (s nimi zpravidla nepotřebujeme pracovat).

V případě disjunktivní normální formy je každému pravidlu $\mathcal{R}_i$ přiřazena formule ve tvaru konjunkce:

$$\boldsymbol{A}_i(x) \,\mathbf{\&}\, \boldsymbol{B}_i(y) \tag{2.5}$$

a celému jazykovému popisu normální forma ve tvaru DNF:

$$\boldsymbol{DNF}(x,y) = \bigvee_{i=1}^{m} \big(\boldsymbol{A}_i(x) \,\mathbf{\&}\, \boldsymbol{B}_i(y)\big) \tag{2.6}$$

nebo se pravidla iterpretují implikativním způsobem:

$$\boldsymbol{A}_i(x) \Rightarrow \boldsymbol{B}_i(y) \tag{2.7}$$

a celému popisu je pak přiřazena relace ve tvaru konjunktivní normální formy:

$$\boldsymbol{CNF}(x,y) = \bigwedge_{i=1}^{m} \big(\boldsymbol{A}_i(x) \Rightarrow \boldsymbol{B}_i(y)\big), \tag{2.8}$$

přičemž v obou případech jsou formule $\boldsymbol{A}_i(x)$, $i = 1, \ldots, m$ přiřazeny antecedentům pravidel $\mathcal{R}_i$ (výrazům „X is $\mathscr{A}_i$“) a $\boldsymbol{B}_i$ jejich konsekventům (tj. výrazům „Y is $\mathscr{B}_i$“).

Na sémantické úrovni je nejprve každé formuli $\boldsymbol{A}_i(x)$ přiřazena vhodná fuzzy množina $\tilde{A}_i \subsetneq U$ a každé formuli $\boldsymbol{B}_i$ vhodná fuzzy množina $\tilde{B}_i \subsetneq V$, přičemž fuzzy množiny $\tilde{A}_i$ se snažíme volit tak, aby tvořily fuzzy pokrytí definičního oboru funkce f.

Každé pravidlo $\mathcal{R}_i$ nějakým způsobem charakterizuje chování funkce f na části kartézského součinu $U \times V$, na sémantické úrovni je mu tedy přiřazena fuzzy relace $R_i \subsetneq U \times V$. Způsob vytvoření této fuzzy relace vychází z tvaru formule (2.6), resp. (2.8) a je dán interpretací spojek $\mathbf{\&}$ resp. $\Rightarrow$. Spojka $\mathbf{\&}$ se obvykle interpretuje pomocí vhodně zvolené t-normy (viz poznámka 1) t a obdobně spojka $\Rightarrow$ jako odpovídající operace rezidua $\xrightarrow{t}$ ze zvolené algebry pravdivostních hodnot.

Tímto postupem získáme ke každému pravidlu odpovídající fuzzy relace R_i (viz obr. 2.3), jejichž agregováním získáme konečnou interpretaci normálních forem (2.6), resp. (2.8). Při konjunktivní interpretaci pravidel dostáváme výslednou relaci ve tvaru:

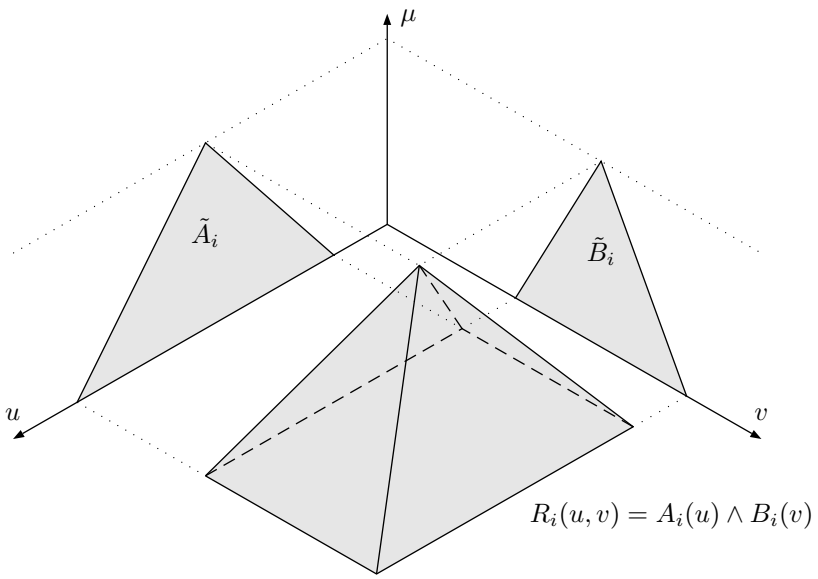
$$R_{\mathrm{DNF}}(u,v) = \bigvee_{i=1}^{m} \big(A_i(u) \;t\; B_i(v)\big) \tag{2.9}$$

a pro implikativní interpretaci pravidel dostáváme fuzzy relaci odpovídající konjunktivní normální formě (2.8) ve tvaru:

$$R_{\mathrm{CNF}}(u,v) = \bigwedge_{i=1}^{m} \big(\boldsymbol{A}_i(u) \xrightarrow{t} \boldsymbol{B}_i(v)\big). \tag{2.10}$$

Na obr. 2.4 je naznačen postup inference v případě jednoho pravidla pro ostrý vstup u_0. Výsledná fuzzy množina $\tilde{B}'_i$ je obecně získána jako obraz vstupní fuzzy množiny $\tilde{A}'_i$ ve fuzzy relaci R_i:

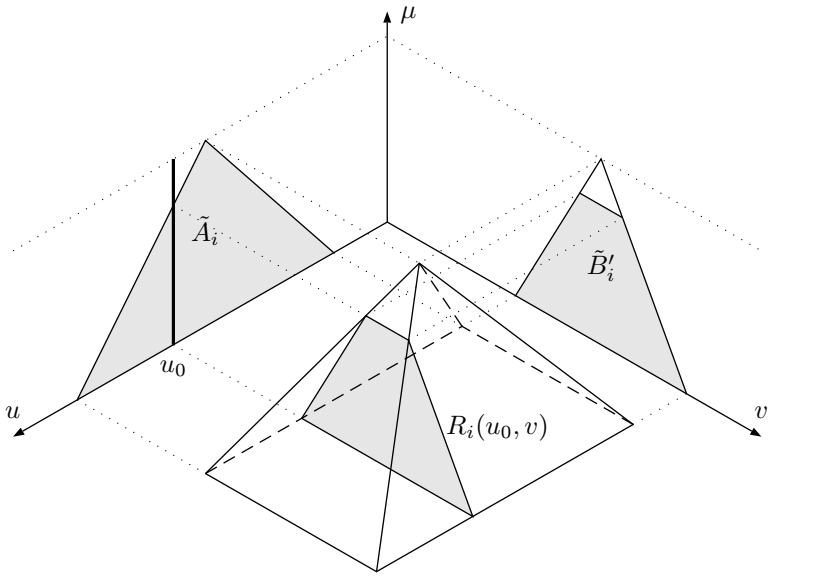
$$\tilde{B}'_i = R_i \circ \tilde{A}'_i. \tag{2.11}$$



Obrázek 2.3: Konstrukce fuzzy relace interpretující konjunktivní pravidlo R_i s použitím minimové t-normy.

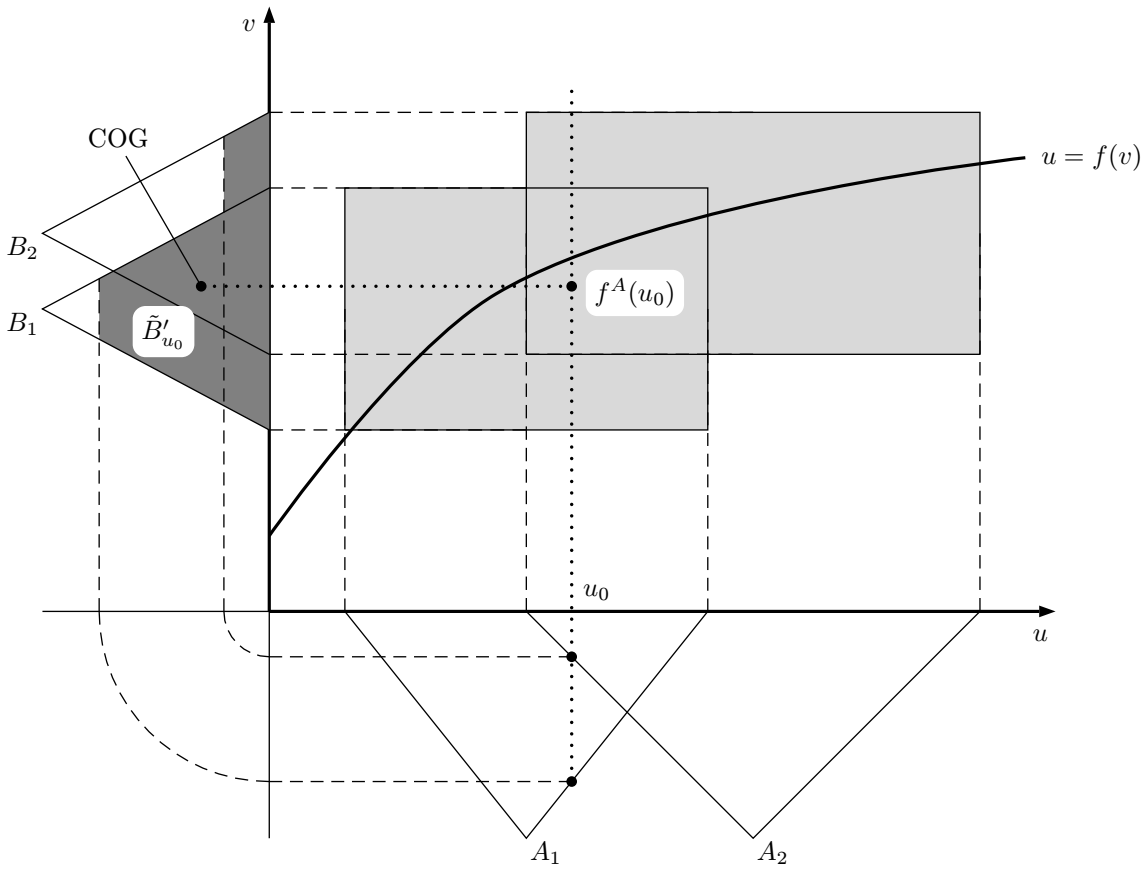
V případě ostrého vstupu je situace jednodušší a výsledná fuzzy množina je určena přímo řezem relace R_i rovinou $\sigma = u_0$:

$$B_i'(v) = R_i(u_0, v). \tag{2.12}$$



Obrázek 2.4: Inference se vstupním singletonem.

Postup uvedený na obr. 2.5 se v praxi používá nejčastěji a nazývá se *Mamdani-Assilianova metoda*. Postup vyhodnocení výstupu pro zadaný vstup u_0 je následující. Zadané vstupní hodnotě u_0 odpovídá výstupní fuzzy množina $\tilde{B}'_{u_0}$, kterou získáme sjednocením fuzzy množin $\tilde{B}_1$, $\tilde{B}_2$ omezených hodnotami stupňů příslušnosti $A_1(u_0)$ a $A_2(u_0)$. Konečný obraz $v_0 = f^A(u_0)$ získáme defuzzifikací množiny $\tilde{B}'_{u_0}$ metodou COG [17].



Obrázek 2.5: Mamdani–Assilianova metoda aproximace funkce f.

2.2 Takagi-Sugeno pravidla

Pravidla typu Takagi-Sugeno (TS-pravidla) jsou speciálním typem fuzzy IF–THEN pravidel sloužících k popisu funkce. Jedná se o podmíněná pravidla, ve kterých je hodnota sukcedentové proměnné Y vyjádřena přesným lineárním vztahem antecedentových proměnných. Pravidla mají tedy následující tvar:

$$\mathcal{R} : \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathscr{A}_n\ \mathsf{THEN}\ Y = b_0 + b_1X_1 + \cdots + b_nX_n. \tag{2.13}$$

Lze také uvažovat speciální případ, kdy pravá strana pravidla je tvořena jediným číslem, tedy:

$$\mathcal{R} : \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathscr{A}_n\ \mathsf{THEN}\ Y = b. \tag{2.14}$$

TS-pravidla se interpretují pouze na sémantické úrovni. Předpokládejme, že máme dánu soustavu m pravidel ve tvaru (2.13) a že jsou dána univerza $U_1, \ldots, U_n$ nezávislých proměnných a univerzum V závislé proměnné. Aproximovaná funkce je tedy tvaru: $f : U_1 \times \cdots \times U_n \longrightarrow V$. Dále je ke každému výrazu $\mathscr{A}_i$ přiřazena vhodná fuzzy množina $\tilde{A}_i \subsetneq U_i$ a spojce AND je přiřazena nějaká t-norma (nejčastěji minimum).

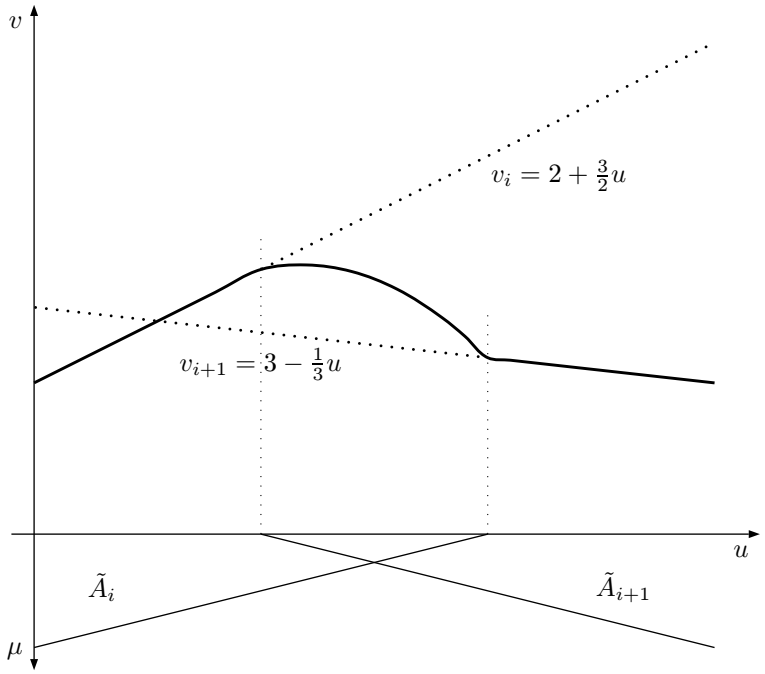
Aproximační funkci f^A pak dostaneme pomocí vztahu:

$$f^A(u_1,\dots,u_n) = \frac{\sum_{i=1}^{m} \big(A_{i1}(u_1) \wedge \cdots \wedge A_{in}(u_n)\big) \cdot (b_{i0} + b_{i1}u_1 + \cdots + b_{in}u_n)}{\sum_{i=1}^{m} \big(A_{i1}(u_1) \wedge \cdots \wedge A_{in}(u_n)\big)}. \tag{2.15}$$

V případě jedné vstupní proměnné dostaneme následující jednodušší vztah:

$$f^A(u) = \frac{\sum_{i=1}^{m} A_i(u) \cdot (b_{i0} + b_{i1}u)}{\sum_{i=1}^{m} A_i(u)}. \tag{2.16}$$

Pro ilustraci je možno vidět odpovídající aproximační funkci jedné proměnné schématicky znázorněnu na obrázku 2.6.



Obrázek 2.6: Aproximace funkce vytvořená na základě TS-pravidel.

Co se týče vlastností výsledné aproximační funkce f^A, je možno velmi zhruba říci, že v místech, kde nedochází k „překryvu“ fuzzy množin tvořících pokrytí univerza, je průběh aproximační funkce lineární, přičemž v místech vzájemného překrývání fuzzy množin $\tilde{A}_i$, $\tilde{A}_{i+1}$ dochází k plynulému napojení jedné lineární funkce na druhou. Důležitou vlastností metody aproximace pomocí TS-pravidel je, že u ní není zapotřebí defuzzifikační procedura.

2.3 Fuzzy transformace

Třetí důležitou metodou fuzzy aproximace je *fuzzy transformace*, často označovaná zkráceně jako F-transformace. Fuzzy transformace díky svým vlastnostem nachází uplatnění v nejrůznějších

oborech, jako např. filtrace signálu pro odstranění šumu [44], komprese obrázků [14, 45], numerické řešení diferenciálních rovnic [48, 49, 50], hledání závislostí mezi daty aj.

Základní myšlenka F-transformace jako aproximační techniky spočívá v nahrazení spojité funkce její diskrétní reprezentací použitím přímé F-transformace. Poté je tato diskrétní reprezentace transformována zpět do prostoru spojitých funkcí použitím inverzní F-transformace. Výsledek získaný aplikací dopředné a zpětné F-transformace je dobrou aproximací původní funkce v tom smyslu, že podle dále uvedené věty 3 je možno získat aproximaci s libovolně nízkou chybou aproximace.

2.3.1 Jednorozměrná fuzzy transformace

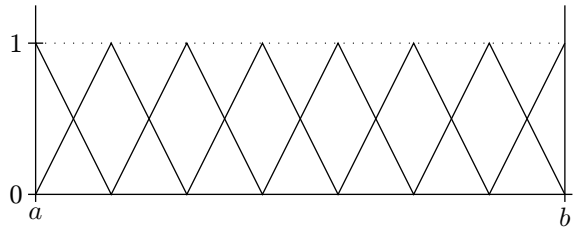
Definice 10 (Fuzzy rozklad intervalu, bázické funkce).
Nechť je dán interval $[a;b] \in \mathbb{R}$, kde $a < b$ a dále $n \geq 2$ jeho vnitřních bodů $x_1 < \cdots < x_n$, nazývaných *uzly* fuzzy rozkladu, přičemž $x_1 = a$, $x_n = b$. Systém fuzzy množin $\mathcal{P} = \{\tilde{A}_i \mid \tilde{A}_i \subsetneq [a;b]\}$ tvoří *fuzzy rozklad intervalu* $[a;b]$, pokud pro funkce příslušnosti A_k odpovídajících fuzzy množin pro každé $k = 1, \ldots, n$:

1. $A_k : [a;b] \to [0;1]$, $A_k(x_k) = 1$
2. $A_k(x) = 0$, pokud $x \notin [x_{k-1}; x_{k+1}]$, kde $x_0 = a$ a $x_{n+1} = b$
3. $A_k(x)$ je spojitá funkce
4. $A_k(x)$, $k = 2, \ldots, n$ monotónně roste na $[x_{k-1}; x_k]$ a $A_k(x)$ pro $k = 1, \ldots, n-1$ monotónně klesá na $[x_k; x_{k+1}]$
5. Ruspiniho podmínka:
$$\sum_{k=1}^{n} A_k(x) = 1, \qquad x \in [a;b]$$
Řekneme, že fuzzy rozklad $\tilde{A}_i, \ldots, \tilde{A}_n$ je *rovnoměrný*, jestliže uzly $x_1, \ldots, x_n$ jsou ekvidistantní, tj. $x_k = a + h(k-1), k = 1, \ldots, n$, kde $h = (b-a)/(n-1)$ a dále je splněno pro všechna $k = 2, \ldots, n-1$:
6. $A_k(x_k - x) = A_k(x_k + x)$ pro všechna $x \in [0;h]$
7. $A_k(x) = A_{k-1}(x-h)$ pro všechna $x \in [x_k; x_{k+1}]$ a $A_{k+1}(x) = A_k(x-h)$ pro všechna $x \in [x_k; x_{k+1}]$
Jinak říkáme, že fuzzy rozklad je *obecný*.

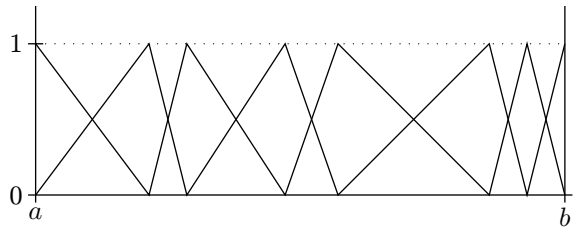
Funkce příslušnosti $A_k, k = 1, \ldots, n$ budeme v souvislosti s F-transformací nazývat *bázické funkce*.

Definice 11 (Fuzzy transformace).
Mějme dánu spojitou reálnou funkci f definovanou na intervalu $[a;b]$ společně s jeho fuzzy rozkladem $\mathcal{P} = \{\tilde{A}_1, \ldots, \tilde{A}_k\}$. Potom můžeme definovat *(integrální) F-transformaci* funkce f vzhledem k $\mathcal{P}$ jako k-tici reálných čísel $[F_1, \ldots, F_k]$ tzv. *komponent* vypočtených podle vztahu:

$$F_i = \frac{\int_a^b f(x) A_i(x)\,\mathrm{d}x}{\int_a^b A_i(x)\,\mathrm{d}x} \qquad \text{pro } i = 1, \ldots, k, \tag{2.17}$$



Obrázek 2.7: Rovnoměrný fuzzy rozklad intervalu $[a; b]$.



Obrázek 2.8: Obecný (nerovnoměrný) fuzzy rozklad intervalu $[a; b]$.

kterou značíme $\mathbf{F}_{\mathcal{P}}[f]$ a máme tedy:

$$\mathbf{F}_{\mathcal{P}}[f] = [F_1, \ldots, F_k], \tag{2.18}$$

přičemž komponenty $F_2, \ldots, F_{k-1}$ nazýváme *regulární* a komponenty F_1, F_k *singulární*.

V praxi má mnohem širší uplatnění *diskrétní F-transformace*, kterou používáme v případech, kdy funkci známe pouze v několika bodech.

Definice 12 (Diskrétní F-transformace).
Mějme dánu (diskrétní) funkci f v konečném počtu bodů $P = \{p_1, \ldots, p_N\} \subset [a; b]$ a fuzzy rozklad $\mathcal{P} = \{\tilde{A}_1, \ldots, \tilde{A}_k\}$ intervalu $[a; b]$. Potom, pokud je množina P *dostatečně hustá* vzhledem k fuzzy rozkladu $\mathcal{P}$, což je dáno podmínkou[2]:

$$\bigwedge_{i=1}^{k} \bigvee_{j=1}^{N} A_i(p_j) > 0, \tag{2.19}$$

můžeme definovat diskrétní F-transformaci funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$ jako k-tici reálných komponent $\mathbf{F}_{\mathcal{P}}[f] = [F_1, \ldots, F_k]$ vypočtených podle:

$$F_i = \frac{\sum_{j=1}^{N} f(p_j) A_i(p_j)}{\sum_{j=1}^{N} A_i(p_j)} \qquad \text{pro } i = 1, \ldots, k. \tag{2.20}$$

Poznámka 4.
Pro smysluplné využití by mělo navíc platit, aby $N \gg k$.

[2]Podmínka (2.19) zajišťuje, že každá fuzzy množina rozkladu P musí obsahovat alespoň jeden bod s nenulovým stupněm příslušnosti.

Poznámka 5.
Výše uvedené definice F-transformace (2.17) a (2.20) někdy nazýváme *dopředná* F-transformace, protože pro získání výsledné aproximační funkce musíme ještě definovat *zpětnou*, resp. *inverzní* F-transformaci $f^{(-1)}_{\mathcal{P},\mathbf{F}}$.

Definice 13 (Inverzní F-transformace).
Mějme dán fuzzy rozklad $\mathcal{P} = \{\tilde{A}_1, \ldots, \tilde{A}_k\}$ intervalu $[a; b]$ společně s vektorem komponent $\mathbf{F} = [F_1, \ldots, F_k] \in \mathbb{R}^k$. Pak funkce definovaná jako

$$f^{(-1)}_{\mathcal{P},\mathbf{F}}(x) = \sum_{i=1}^{k} F_i A_i(x) \tag{2.21}$$

se nazývá *inverzní F-transformace.*

Definice 14 (Inverzní F-transformace funkce).
Mějme dánu spojitou reálnou funkci f definovanou na intervalu $[a; b]$ společně s jeho fuzzy rozkladem $\mathcal{P} = \{\tilde{A}_1, \ldots, \tilde{A}_k\}$. Potom můžeme definovat *(integrální) inverzní F-transformaci* funkce f vzhledem k $\mathcal{P}$ jako:

$$f^{(-1)}_{\mathcal{P},f}(x) = f^{(-1)}_{\mathcal{P},\mathbf{F}_{\mathcal{P}}[f]}(x), \tag{2.22}$$

kde $\mathbf{F}_{\mathcal{P}}[f]$ představuje vektor komponent dopředné F-transformace funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$, jehož složky jsou vypočítány pomocí vztahu (2.17).

Definice 15 (Inverzní F-transformace diskrétní funkce).
Mějme dánu (diskrétní) funkci f v konečném počtu bodů $P = \{p_1, \ldots, p_N\} \subset [a; b]$ a fuzzy rozklad $\mathcal{P} = \{\tilde{A}_1, \ldots, \tilde{A}_k\}$ intervalu $[a; b]$. Potom, pokud je množina P dostatečně hustá vzhledem k fuzzy rozkladu $\mathcal{P}$ (viz podmínka (2.19)), můžeme definovat inverzní F-transformaci funkce f jako:

$$f^{(-1)}_{\mathcal{P},f}(x) = f^{(-1)}_{\mathcal{P},\mathbf{F}_{\mathcal{P}}[f]}(x), \tag{2.23}$$

kde $\mathbf{F}_{\mathcal{P}}[f]$ představuje vektor komponent diskrétní F-transformace funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$, jehož složky jsou vypočítány pomocí vztahu (2.20).

Výslednou aproximační funkci f^A funkce f za použití F-transformace můžeme vyjádřit jako

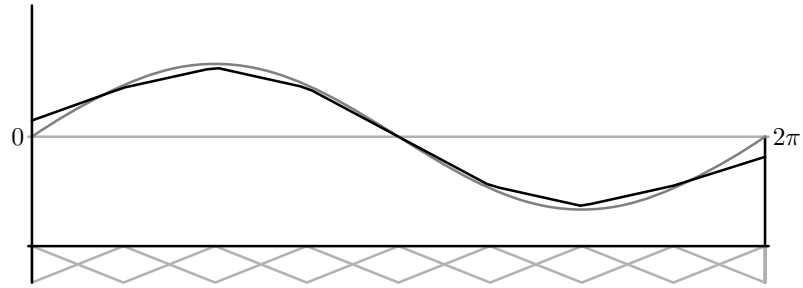
$$f^A(x) = f^{(-1)}_{\mathcal{P},f}(x) \tag{2.24}$$

Podle toho, zda je funkce f spojitá, resp. diskrétní, je při výpočtu použito vztahu (2.22), resp. (2.23).

2.3.2 Vícerozměrná fuzzy transformace

Především z praktických důvodů je vhodné právě uvedenou metodu rozšířit pro vícerozměrný případ. Existuje více způsobů, jak toto provést. K tomu je nutno vyřešit dva dílčí úkoly. Jednak je nutno provést dělení vícerozměrné oblasti a za druhé určit tvar bázických funkcí nad jednotlivými prvky tohoto dělení.

V krátkosti uvedeme dva možné přístupy zobecnění fuzzy transformace pro vícerozměrné funkce:



Obrázek 2.9: Ukázka aproximace funkce $\sin(x)$ na intervalu $[0; 2\pi]$ při použití rovnoměrného rozkladu o 7 vnitřních uzlech.

- *Kartézská F-transformace* je přímočarým zobecněním vztahů (2.17), (2.20) a (2.21) na jejich vícerozměrnou variantu.

- *Triangulární F-transformace* spočívá v zobecnění fuzzy rozkladu d-rozměrné oblasti za pomoci metody triangularizace.

Kartézská F-transformace

Definice 16 (Kartézský fuzzy rozklad d-rozměrného intervalu).
Mějme dán d-rozměrný interval $I^d \subseteq \mathbb{R}^d$ tvořený kartézským součinem d reálných intervalů $[a_1; b_1] \times [a_2; b_2] \times \cdots \times [a_d; b_d]$. Dále mějme dán pro každý interval I_i jeho fuzzy rozklad sestávající z k_i fuzzy množin $\tilde{A}^i_1, \ldots, \tilde{A}^i_{k_i}$. *Kartézský fuzzy rozklad* je tvořen systémem d-rozměrných fuzzy množin $\tilde{A}_{(i_1 \cdots i_d)}$ o celkovém počtu $\prod_{i=1}^{d} k_i$, jejichž funkce příslušnosti jsou tvořeny součinem bázických funkcí odpovídajících fuzzy rozkladům intervalů v jednotlivých dimenzích:

$$A_{(i_1 \cdots i_d)}(x_1, \ldots, x_d) = A^1_{(i_1)}(x_1) \cdots A^d_{(i_d)}(x_d). \tag{2.25}$$

Řekneme, že kartézský fuzzy rozklad je *rovnoměrný*, pokud každý z fuzzy rozkladů odpovídající intervalům $I_1, \ldots, I_d$ je rovnoměrný. Jinak se jedná o *obecný* fuzzy rozklad.

O důležité vlastnosti právě definovaného kartézského rozkladu hovoří následující věta, jejíž důkaz je uveden v [27].

Věta 1.
Kartézský fuzzy rozklad splňuje Ruspiniho podmínku:

$$(\forall X \in I^d) : \sum_{(i_1 \cdots i_d)} A_{(i_1 \cdots i_d)}(X) = 1, \tag{2.26}$$

Definice 17 (Kartézská F-transformace).
Mějme dánu d-rozměrnou spojitou reálnou funkci f definovanou na kartézském součinu d reálných intervalů: $\mathcal{D}(f) = [a_1; b_1] \times [a_2; b_2] \times \cdots \times [a_d; b_d]$. Dále mějme dán kartézský fuzzy rozklad $\mathcal{P}$ intervalu $\mathcal{D}(f)$.

Pak můžeme definovat (integrální) kartézskou F-transformaci funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$ jako d-rozměrnou matici reálných čísel $\mathbf{F}_{\mathcal{P}} = [F_{(i_1\cdots i_d)}]$ vypočtených podle:

$$F_{(i_1\cdots i_d)} = \frac{\int_{a_d}^{b_d}\cdots\int_{a_1}^{b_1} f(x_1,\ldots,x_d)\, A_{(i_1\cdots i_d)}(x_1,\ldots,x_d)\ \mathrm{d}x_1\cdots\mathrm{d}x_d}{\int_{a_d}^{b_d}\cdots\int_{a_1}^{b_1} A_{(i_1\cdots i_d)}(x_1,\ldots,x_d)\ \mathrm{d}x_1\cdots\mathrm{d}x_d}, \tag{2.27}$$

resp. po rozepsání:

$$F_{(i_1\cdots i_d)} = \frac{\int_{a_d}^{b_d}\cdots\int_{a_1}^{b_1} f(x_1,\ldots,x_d)\, A^1_{(i_1)}(x_1)\cdots A^d_{(i_d)}(x_d)\ \mathrm{d}x_1\cdots\mathrm{d}x_d}{\int_{a_d}^{b_d}\cdots\int_{a_1}^{b_1} A^1_{(i_1)}(x_1)\cdots A^d_{(i_d)}(x_d)\ \mathrm{d}x_1\cdots\mathrm{d}x_d}. \tag{2.28}$$

Pro komponenty přímé d-rozměrné diskrétní F-transformace dostaneme vztah:

$$F_{(i_1\cdots i_d)} = \frac{\sum_{j=1}^{N} f(p_j) A_{(i_1\cdots i_d)}(p_j)}{\sum_{j=1}^{N} A_{(i_1\cdots i_d)}(p_j)}, \tag{2.29}$$

kde $p_j \in \mathcal{D}$ jsou body, ve kterých je známa hodnota funkce f. Pro zpětnou (inverzní) d-rozměrnou F-transformaci dostaneme:

$$f^{(-1)}_{\mathcal{P},\mathbf{F}}(X) = \sum_{(i_1\cdots i_d)} F_{(i_1\cdots i_d)} A_{(i_1\cdots i_d)}(X). \tag{2.30}$$

Triangulární F-transformace

Pokud máme aproximovat funkci definovanou na obecné oblasti Ω, která není tvořena d-rozměrným kvádrem, můžeme buďto tuto oblast vnořit do nějakého kvádru a dále postupovat pomocí dříve uvedené kartézské F-transformace, nebo provést triangularizaci dané oblasti a dále použít následující postup tzv. *triangulární F-transformace*. V dalším se pro jednoduchost při výkladu omezíme na rovinu, takže budeme dále hovořit o dvojrozměrné triangulární F-transformaci, přičemž v závěru nastíníme zobecnění tohoto postupu pro d-rozměrný případ.

Definice 18 (Triangularizace, triangulární rozklad).
Nechť $\Omega \subseteq \mathbb{R}^2$ je polygonální oblast[3] ležící v rovině α. Triangularizací dané oblasti budeme rozumět strukturu $\mathcal{T}\langle V, T\rangle$, kde V je množina vrcholů triangularizace $V = \{V_i\}$ ležící v rovině α a $T = \{t_j\}$ je množina trojúhelníků jejichž vrcholy jsou z množiny V, pokud platí:

- libovolné dva navzájem různé trojúhelníky z množiny T se vzájemně protínají nejvýše v jednom společném vrcholu nebo v jedné společné hraně:

$$(\forall s,t \in T),\ s \neq t,\ s = \triangle(A,B,C):\ s \cap t \in \big\{\emptyset, \{A\}, \{B\}, \{C\}, \{|AB|\}, \{|BC|\}, \{|CA|\}\big\}$$

- každý bod z množiny V je vrcholem alespoň jednoho trojúhelníka:

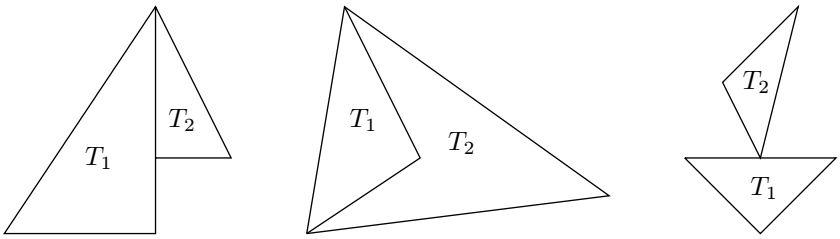
$$(\forall M \in V)\ (\exists t \in T),\ t = \triangle(A,B,C):\ M \in \{A,B,C\}$$

[3] Obecně nemusí být tato oblast konvexní a může obsahovat „díry“.

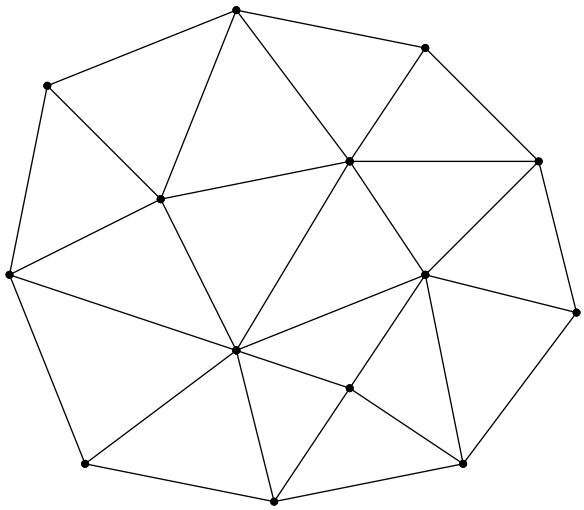
- sjednocení všech trojúhelníků z množiny T přesně pokrývá danou oblast Ω:

$$\bigcup_{i=i}^{|T|} t_i = \Omega$$

Obrázek 2.10 ukazuje příklady vzájemných poloh trojúhelníků, které tato definice nepovoluje. Na obrázku 2.11 je naopak vidět typický příklad provedené triangularizace.



Obrázek 2.10: Protipříklady poloh trojúhelníků odporujících definici triangularizace.



Obrázek 2.11: Příklad triangularizace oblasti.

Vyjděme nyní z předpokladu, že máme dánu triangularizaci $\mathcal{T}$ dané oblasti Ω a půjde nám o vytvoření fuzzy rozkladu na základě této triangularizace.

Definice 19 (Triangulární fuzzy rozklad).
Mějme dánu triangularizaci $\mathcal{T}$ dané oblasti Ω. Triangulární fuzzy rozklad oblasti Ω nazveme systém fuzzy množin $\mathcal{P} = \{\tilde{A}_i \subsetneq \Omega\}$ zkonstruovaný následovně:

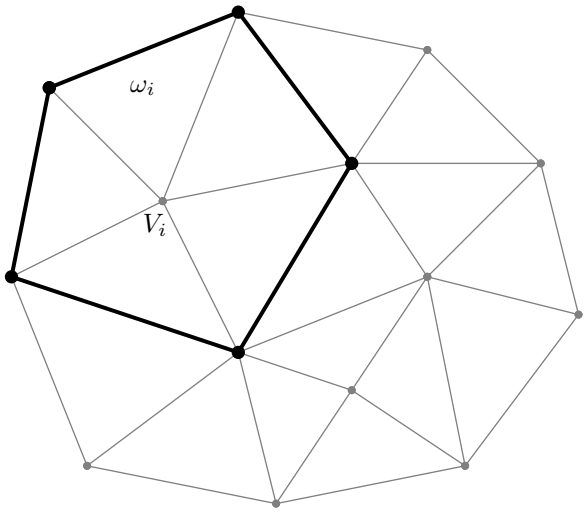
pro každý vrchol $V_i \in V$ dané triangularizace sestrojíme fuzzy množinu $\tilde{A}_i$, jejímž supportem je oblast ω_i, která vznikne sjednocením všech trojúhelníků triangularizace se společným vrcholem V_i.

$$\mathrm{Supp}(\tilde{A}_i) = \omega_i$$

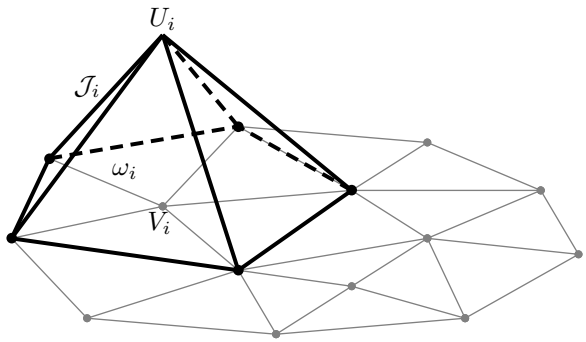
$$\omega_i = \bigcup_j \{t_j \in T \mid V_i \in t_j\}, \quad \text{pro } i = 1, \ldots, |V|$$

Výslednou bázickou funkci A_i sestrojíme ve tvaru pláště jehlanu $\mathcal{J}_i$ s vrcholem v bodě U_i a podstavou ω_i, přičemž bod U_i získáme z bodu V_i zvednutím z původní roviny α triangulované oblasti kolmo vzhůru do hladiny s hodnotou 1.

$$\begin{aligned} \text{graf}(A_i) &= \mathcal{J}_i(U_i, \omega_i) \\ &\leftrightarrow U_iV_i \perp \alpha \\ &|U_iV_i| = 1 \end{aligned}$$



Obrázek 2.12: Konstrukce oblasti ω_i k bodu V_i.



Obrázek 2.13: Konstrukce fuzzy množiny nad oblastí ω_i.

Věta 2.
Triangulární fuzzy rozklad splňuje Ruspiniho podmínku, tedy platí:

$$(\forall X \in \Omega) : \sum_{i=1}^{|V|} A_i(X) = 1.$$

Důkaz 1. Pokud bod X splývá s některým vrcholem V_i z vrcholů množiny V, je z konstrukce fuzzy rozkladu zřejmé, že existuje právě jedna fuzzy množina obsahující daný bod s nenulovým stupněm příslušnosti, přičemž je tento stupeň roven 1.

V ostatních případech budeme k provedení důkazu potřebovat vyjádření hodnoty funkce příslušnosti A_i v libovolném bodě X.

Uvažujme fuzzy množinu $\tilde{A}_i$ sestrojenou při vrcholu V_i a předpokládejme, že bod X padne do trojúhelníku $\triangle t(V_i, V_j, V_k)$. Funkce příslušnosti A_i má nad tímto trojúhelníkem tvar roviny β procházející body $[V_i^x, V_i^y, 1]$, $[V_j^x, V_j^y, 0]$ a $[V_k^x, V_k^y, 0]$.

Z toho dostáváme pro hodnotu funkce příslušnosti $A_i(X)$ vztah odpovídající hodnotě z-tové souřadnice bodu ležícího v rovině β, jehož kolmým průmětem do roviny xy je bod X. Máme tedy:

$$A_i(X) = \frac{\begin{vmatrix} X - V_j \\ V_k - V_j \end{vmatrix}}{\begin{vmatrix} V_i - V_j \\ V_k - V_j \end{vmatrix}},$$

kde $\begin{vmatrix} \vec{u} \\ \vec{v} \end{vmatrix}$ je zkratka pro zápis determinantu $\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}$.

Je zřejmé, že pro Ruspiniho podmínku hrají významnou roli v podobě nenulových hodnot stupňů příslušnosti nejvýše tři fuzzy množiny podle toho, zda bod X splývá s některým vrcholem triangularizace, nebo zda leží na hraně, či uvnitř některého trojúhelníka triangularizace. Všechny tyto tři případy jsou zahrnuty v následujícím vztahu, ve kterém opět předpokládáme, že bod $X \in \triangle t(V_i, V_j, V_k)$:

$$\sum_{l=1}^{|V|} A_l(X) = A_i(X) + A_j(X) + A_k(X) = \frac{\begin{vmatrix} X - V_j \\ V_k - V_j \end{vmatrix}}{\begin{vmatrix} V_i - V_j \\ V_k - V_j \end{vmatrix}} + \frac{\begin{vmatrix} X - V_k \\ V_i - V_k \end{vmatrix}}{\begin{vmatrix} V_j - V_k \\ V_i - V_k \end{vmatrix}} + \frac{\begin{vmatrix} X - V_i \\ V_j - V_i \end{vmatrix}}{\begin{vmatrix} V_k - V_i \\ V_j - V_i \end{vmatrix}},$$

po úpravě[4] pravé strany konečně dostáváme

$$\sum_{l=1}^{|V|} A_l(X) = 1.$$

□

Definice 20 (Triangulární F-transformace).
Mějme dánu spojitou reálnou funkci $f(x, y)$ definovanou na polygonální oblasti Ω společně s triangulárním fuzzy rozkladem $\mathcal{P} = \{\tilde{A}_i\}$ sestrojeným nad triangularizací $\mathcal{T}\langle V, T\rangle$.

(Integrální) triangulární F-transformace funkce f vzhledem k rozkladu $\mathcal{P}$ nazveme k-tici reálných čísel $\mathbf{F}_\mathcal{P} = [F_1, \ldots, F_k]$ ($k = |V|$) vypočtených podle vztahu:

[4] Ke zjednodušení výrazu na pravé straně byl použit programový nástroj YACAS pro symbolickou manipulaci s matematickými výrazy.

$$F_i = \frac{\int_\Omega f(x,y)A_i(x,y)\ \mathrm{d}x\,\mathrm{d}y}{\int_\Omega A_i(x,y)\ \mathrm{d}x\,\mathrm{d}y} \qquad \text{pro } i = 1,\dots,k. \tag{2.31}$$

Komponenty diskrétní triangulární F-transformace získáme pomocí:

$$F_i = \frac{\sum_{j=1}^{N} f(x_j,y_j)A_i(x_j,y_j)}{\sum_{j=1}^{N} A_i(x_j,y_j)} \qquad \text{pro } i = 1,\dots,k, \tag{2.32}$$

kde N je počet bodů, ve kterých známe funkční hodnoty funkce f. Pro inverzní triangulární F-transformaci máme vztah:

$$f^{(-1)}_{\mathcal{P},\mathbf{F}}(x,y) = \sum_{i=1}^{k} F_i A_i(x,y). \tag{2.33}$$

Poznámka 6.
Uvedený postup triangulární F-transformace je možno zobecnit na d-rozměrný případ. Toto zobecnění spočívá především v zobecnění fuzzy rozkladu dané oblasti pomocí d-rozměrné triangularizace. S přibývajícími rozměry se zvyšuje i rozměr jednotlivých segmentů, na které je daná oblast rozdělena. Zatímco u dvojrozměrné triangularizace to jsou trojúhelníky, v případě trojrozměrné triangularizace se jedná o čtyřstěny a obecně pracujeme se simplexy vyššího stupně v případě vícerozměrných prostorů. Konstrukce d-rozměrných fuzzy množin nad takovýmto dělením spolu s výpočtem komponent fuzzy transformace a inverzní F-transformace je analogická jako v již uvedeném dvojrozměrném případě.

2.3.3 Shrnutí

Co se týče vztahů pro výpočet komponent F-transformace i inverzní F-transformace, můžeme všechny výše uvedené varianty shrnout do následujících obecných předpisů:

Předpokládejme, že máme dánu spojitou funkci f, jejímž definičním oborem je d-rozměrná oblast $\mathcal{D}(f) = \Omega \subseteq \mathbb{R}^d$, pro kterou máme dán fuzzy rozklad $\mathcal{P}$ o k fuzzy množinách $\tilde{A}_1,\dots,\tilde{A}_k$. Potom máme:

- **(Integrální) F-transformace** funkce $f(X)$ vzhledem k rozkladu $\mathcal{P}$ je k-tice reálných čísel $\mathbf{F}_\mathcal{P} = [F_1,\dots,F_k]$ vypočtených pomocí:

$$F_i = \frac{\int_\Omega f(X)A_i(X)\ \mathrm{d}X}{\int_\Omega A_i(X)\ \mathrm{d}X} \qquad \text{pro } i = 1,\dots,k \tag{2.34}$$

- **Diskrétní F-transformace** funkce $f(X)$ vzhledem k rozkladu $\mathcal{P}$ je k-tice reálných čísel $\mathbf{F}_\mathcal{P} = [F_1,\dots,F_k]$ vypočtených pomocí:

$$F_i = \frac{\sum_{j=1}^{N} f(X_j)A_i(X_j)}{\sum_{j=1}^{N} A_i(X_j)} \qquad \text{pro } i = 1,\dots,k \tag{2.35}$$

- **Inverzní F-transformace**

$$f^{(-1)}_{\mathcal{P},\mathbf{F}}(X) = \sum_{i=1}^{k} F_i A_i(X) \tag{2.36}$$

2.4 Výpočetní složitost F-transformace

V této části se budeme zabývat výpočetní složitostí algoritmů souvisejících s F-transformací. V praktických úlohách, i když máme funkci f, kterou chceme pomocí F-transformace aproximovat, danou exaktním analytickým předpisem, ve většině případů místo integrální (spojité) F-transformace používáme přibližné řešení získané pomocí diskrétní F-transformace s vhodně zvolenou hustotou rozložení reprezentativních dat v definičním oboru $\mathcal{D}(f)$ dané funkce.

Vzhledem k tomuto faktu hraje klíčovou úlohu algoritmus DFT pro výpočet komponent diskrétní F-transformace. Druhým algoritmem je algoritmus IFT pro výpočet hodnoty zpětné (inverzní) F-transformace $f^{(-1)}_{\mathcal{P},\mathbf{F}}$ v obecném bodě $X \in \mathcal{D}(f)$.

Jak uvidíme dále, konečná výpočetní složitost navržených obecných algoritmů velmi závisí na typu řešených úloh a vlastnostech vstupních parametrů, zejména na fuzzy rozkladu definičního oboru.

Pro odhady složitostí jednotlivých algoritmů budeme využívat zažité konvence značení pomocí $\mathcal{O}(g)$ definované následovně.

Definice 21 (Neostrý horní odhad).
Mějme dány funkce f a g definované na oboru přirozených čísel. Říkáme, že funkce g je neostrým horním odhadem funkce f, pokud platí:

$$(\exists n_0 > 0)(\exists k > 0)(\forall n > n_0) : f(n) \leq kg(n),$$

což značíme pomocí $f \in \mathcal{O}(g)$.

Definice 22 (Složitost algoritmu).
Složitost $\mathcal{T}(A)$ algoritmu A definujeme jako funkci $\mathcal{T}_A(n)$ určující počet operací nutných k jeho vykonání v závislosti na velikosti vstupních dat v nejhorším možném případě. Nechť $T_A(d)$ značí počet operací nutných k provedení konkrétního výpočtu algoritmu nad daty d, potom můžeme psát:

$$\mathcal{T}_A(n) = \bigvee_{|d|=n} T_A(d),$$

kde pomocí $|d|$ značíme velikost vstupních dat. Nejčastěji pracujeme s neostrými horními odhady složitosti algoritmu, takže například říkáme, že složitost algoritmu je v $\mathcal{O}(n^2)$, čímž míníme, že algoritmus má kvadratickou složitost.

Algoritmus diskrétní F-transformace

Pro všechny varianty dříve uvedených diskrétních F-transformací můžeme navrhnout jednotný algoritmus pro výpočet jejich komponent vycházející ze vztahu (2.35). Výpis algoritmu DFT (alg. 1) je nejobecnějším postupem, který budeme dále používat pro další analýzy.

Algoritmus 1: Dopředná diskrétní F-transformace

označení : DFT

vstup : $\left\{\langle X_i; y_i\rangle\right\}_1^N$ – soubor N bodů $X_i \in \mathbb{R}^d$ s funkční hodnotou $y_i = f(X_i)$,
$\mathcal{P} = \left\{\tilde{A}_j\right\}_1^k$ – systém fuzzy množin – fuzzy rozklad definičního oboru $\mathcal{D}(f)$

výstup : $[F_1, \ldots, F_k]$ – k-tice reálných komponent

proměnné: $S_1^{fA}, \ldots, S_k^{fA}$ – pomocné pole čitatelů částečných sum $S_j^{fA} = \sum_{l=1}^{i} f(X_l)A_j(X_l)$,

$S_1^A, \ldots, S_k^A$ – pomocné pole jmenovatelů částečných sum $S_j^A = \sum_{j=l}^{i} A_i(X_l)$,

kde i je index aktuálně zpracovávaných dat během výpočtu.

```
// inicializace (vynulování)
```
for $j := 1$ **to** k **do** $S_j^{fA} := S_j^A := 0$;

```
// hlavní cyklus přes všechna vstupní data
```
for $i := 1$ **to** N **do**

```
    // aktualizace hodnot všech komponent ovlivněných bodem X_i
```
    foreach j : $A_j(X_i) > 0$ **do**
        $S_j^{fA} := S_j^{fA} + y_i \cdot A_j(X_i)$;
        $S_j^A := S_j^A + A_j(X_i)$;
    end

end

```
// finalizace
```
for $j := 1$ **to** k **do** $F_j := S_j^{fA}/S_j^A$

return $[F_1, \ldots, F_k]$;

Algoritmus dopředné diskrétní F-transformace předpokládá existenci funkce f definované na známém oboru $\mathcal{D}(f)$ a má za úkol vypočíst komponenty její F-transformace. Celý definiční obor je přitom možno získat jako sjednocení všech částí zadaného fuzzy rozkladu:

$$\mathcal{D}(f) = \bigcup_{i=1}^{k} \mathrm{Supp}(\tilde{A}_i).$$

Na svém vstupu očekává N uspořádaných dvojic $\langle X_i; y_i\rangle$, kde X_i jsou body z d-rozměrného prostoru, ve kterých je známa aproximovaná funkce a y_i jsou její odpovídající funkční hodnoty $f(X_i)$. Dále je nutno jako vstup zadat fuzzy rozklad $\tilde{A}_1, \ldots, \tilde{A}_k$ definičního oboru $\mathcal{D}(f)$ v podobě d-rozměrných fuzzy množin s odpovídajícími funkcemi stupňů příslušnosti $A_1, \ldots, A_k$. Jejich počet k odpovídá velikosti požadovaného výstupu v podobě k komponent $[F_1, \ldots, F_k]$.

Během výpočtu jsou používány pomocné proměnné $S_1^{fA}, \ldots, S_k^{fA}$ a $S_1^A, \ldots, S_k^A$, jejichž hodnoty jsou aktualizovány s postupně zpracovávanými daty. Představují průběžné součty čitatelů $S_j^{fA} = \sum_{l=1}^{i} f(X_l)A_j(X_l)$ a jmenovatelů $S_j^A = \sum_{j=l}^{i} A_i(X_l)$ zlomků vypočítaných na závěr algoritmu pro získání konečných hodnot výsledných komponent ve tvaru:

$$F_j := \frac{S_j^{fA}}{S_j^A} \qquad \text{pro všechna } j = 1, \ldots, k.$$

Zde se skrývá úskalí možnosti, že hodnota F_j není definovaná v případě, že $S_j^A = 0$, což nastane tehdy, když není splněna podmínka (2.19). Tuto situaci můžeme vyřešit buď tak, že algoritmus v takovém případě skončí s chybou (resp. vyvolá příslušnou výjimku), anebo pro odpovídající komponentu F_j vrátí předem domluvenou hodnotu *undef*.

Nejprve popišme význam vyskytujících se proměnných, které budou hrát roli v odhadu výpočetní složitosti diskrétní F-transformace.

- d – rozměr transformované funkce f resp. dimenzionalita úlohy. $\left(\mathcal{D}(f) \subseteq \mathbb{R}^d\right)$
- N – počet zpracovávaných dat, ve kterých je známa funkční hodnota funkce f. Každý jednotlivý vstup představuje $d+1$ číselných hodnot, celkově můžeme tedy velikost samotných vstupních dat označit jako $M = N(d+1)$.
- k – počet prvků fuzzy rozkladu definičního oboru $\mathcal{D}(f)$.

V dalším budeme předpokládat, že výpočetní složitost určení stupně příslušnosti prvku X do fuzzy množin fuzzy rozkladu $\tilde{A}_1, \ldots, \tilde{A}_k$ je pro všechny fuzzy množiny stejná a rovna hodnotě a:

$$\mathcal{T}(A_i(X)) = a \qquad \text{pro všechna } i = 1, \ldots, k \text{ a } X \in \mathcal{D}(f). \tag{2.37}$$

Ještě poznamenejme, že v praxi bývá nejčastěji $a \in \mathcal{O}(d)$.

Jak je na první pohled patrné, výpočetní složitost algoritmu diskrétní F-transformace je dána součtem složitostí jeho tří částí – inicializace, hlavního cyklu a finalizace, přičemž je zřejmé, že výpočetní složitost inicializace a finalizace je v $\mathcal{O}(k)$ a je vůči složitosti hlavního cyklu vzhledem k podmínce $N \gg k$ zanedbatelná. Složitost hlavního cyklu je dána součinem počtu dat N a složitostí vnitřního těla cyklu. Kdybychom neměli k dispozici žádné další informace o fuzzy rozkladu $\tilde{A}_1, \ldots, \tilde{A}_k$, museli bychom se spokojit s odhadem jeho složitosti v podobě $\mathcal{O}(ka)$ a dostali bychom jako výsledný odhad časové výpočetní složitosti $\mathcal{T}(\mathrm{DFT}) \in \mathcal{O}(Nka)$.

Pokud však vezmeme v úvahu vlastnosti fuzzy rozkladu, můžeme daný odhad v závislosti na druhu fuzzy rozkladu zlepšit. Základní idea spočívá v tom, že každý jednotlivý bod X_i ovlivní pouze malou část komponent, do jejichž oblasti působnosti[5] padne. Optimalizace tedy spočívá v efektivním nalezení všech fuzzy množin $\tilde{A}_j$ z fuzzy rozkladu, pro které je $A_j(X_i) > 0$.

Počet komponent ovlivněných bodem X_i, jakožto i určení toho, o které se jedná, závisí na vlastnostech samotného fuzzy rozkladu.

Nejprve rozeberme případ kartézského fuzzy rozkladu. Z toho, jak je vytvořen, vyplývá, že obecně bod X_j ovlivní hodnoty nejvýše 2^d komponent. Pro nalezení všech komponent, jejichž hodnoty jsou ovlivněny bodem X_i, stačí nalezení jediné, tzv. *referenční* komponenty[6], přičemž další jsou nalezeny snadno, jako její „sousedé“. Po nalezení referenční komponenty jsme schopni každou další komponentu, jejíž hodnota je ovlivněna bodem X_i, nalézt v čase $\mathcal{O}(d)$. Z vlastnosti kartézského fuzzy rozkladu dále vyplývá, že nalezení indexu odpovídající referenční komponenty je možno provést postupně po jednotlivých dimenzích.

Způsob a hlavně rychlost nalezení indexu odpovídající komponenty v rámci jedné dimenze závisí na tom, zda je fuzzy rozklad v této dimenzi rovnoměrný, či obecný. V případě rovnoměrného

[5] Oblast působnosti komponenty F_i je dána hodnotou $\mathrm{Supp}(\tilde{A}_i)$ a představuje oblast, ve které je hodnota funkce $f^{(-1)}_{\mathcal{P},\mathbf{F}}$ inverzní F-transformace ovlivněna hodnotou komponenty F_i.

[6] Referenční komponenta kartézského fuzzy rozkladu vzhledem k bodu X_i je taková komponenta, jejíž všechny dílčí indexy v jednotlivých dimenzích jsou minimální v rámci všech komponent ovlivněných bodem X_i. Index referenční komponenty odpovídající bodu X_i značíme pomocí $\mathrm{ref}(X_i)$.

fuzzy rozkladu je složitost nalezení indexu odpovídající komponenty v rámci jedné dimenze konstantní. Pro nalezení všech dílčích indexů tak dostaneme odhad $\mathcal{O}(d)$.

V případě obecného fuzzy rozkladu intervalu I_i můžeme nalézt odpovídající index v čase $\mathcal{O}(\log(k_i))$ pomocí metody půlení intervalu, kde k_i je počet uzlů tvořících fuzzy rozklad intervalu I_i. Celkově je tedy odhad časové složitosti nalezení všech dílčích indexů roven $\sum_{i=1}^{d} \log(k_i)$. Vzhledem k tomu, že máme $k = \prod_{i=1}^{d} k_i$ dostaneme:

$$\sum_{i=1}^{d} \log(k_i) = \log\left(\prod_{i=1}^{d} k_i\right) = \log(k).$$

Po zjištění indexů v jednotlivých dimenzích jsme schopni vypočíst globální index hledané referenční komponenty v poli komponent $F_1, \ldots, F_k$ v čase $\mathcal{O}(d)$. Celkově tedy pro nalezení referenční komponenty dostáváme v případě rovnoměrného fuzzy rozkladu odhad časové složitosti $\mathcal{O}(d)$ a pro obecný fuzzy rozklad máme odhad $\mathcal{O}(\log(k) + d)$.

Jak již bylo dříve zmíněno, index každé další komponenty, jejíž hodnotu je třeba aktualizovat daty $\langle X_i; y_i \rangle$, jsme schopni nalézt v čase $\mathcal{O}(d)$. Celkem tedy pro nalezení všech zbývajících $2^d - 1$ komponent máme odhad $\mathcal{O}(d2^d)$. Aktualizace každé komponenty zabere podle (2.37) čas $\mathcal{O}(a)$. Pro složitost těla hlavního cyklu tak dostáváme odhady $\mathcal{O}((a+d)2^d)$ v případě rovnoměrného rozkladu a $\mathcal{O}(\log(k)+(a+d)2^d)$ v případě obecného fuzzy rozkladu, což v praxi vzhledem k faktu, že $k \gg d$ a za předpokladu $a \in \mathcal{O}(d)$, dává zjednodušený odhad $\mathcal{O}(\log(k))$ pro obecný fuzzy rozklad.

V případě triangulárního fuzzy rozkladu můžeme použít stejné myšlenky pro optimalizaci jako při obecném kartézském fuzzy rozkladu. Nejkritičtějším místem v odhadu složitosti při použití triangulárního fuzzy rozkladu je způsob nalezení trojúhelníku (resp. obecně d-rozměrného elementu o $d+1$ vrcholech) generující triangularizace, do kterého padne bod X_i. Označme složitost lokalizace tohoto odpovídajícího trojúhelníka jako $\mathcal{O}(t)$. Hodnota t je závislá na datové struktuře zvolené pro uložení informace o fuzzy rozkladu a zřejmě pro ni minimálně platí $t \geq \log|T|$, resp. $t \geq \log|V|$. Po lokalizaci „zasažené“ oblasti je nutno aktualizovat $d+1$ komponent. Pro tělo hlavního cyklu tak dostáváme odhad složitosti v podobě $\mathcal{O}(t + a(d+1))$.

Doposud zjištěné výsledky můžeme shrnout v tabulce 2.2 s přehledem jednotlivých složitostí pro různé druhy fuzzy rozkladů.

	Kartézský fuzzy rozklad		Triangulární fuzzy rozklad
	rovnoměrný	obecný	
Nalezení první komponenty	$\mathcal{O}(d)$	$\mathcal{O}(\log(k))$	$\mathcal{O}(t)$
Nalezení zbývajících komponent	$\mathcal{O}(d2^d)$	$\mathcal{O}(d2^d)$	$\mathcal{O}(d)$
Aktualizace všech komponent	$\mathcal{O}(a2^d)$	$\mathcal{O}(a2^d)$	$\mathcal{O}(ad)$
Celková složitost	$\mathcal{O}((a+d)2^d)$	$\mathcal{O}(\log(k)+(a+d)2^d)$	$\mathcal{O}(t+ad)$
při $a \in \mathcal{O}(d)$, $k \gg d$	$\mathcal{O}(d2^d)$	$\mathcal{O}(\log(k))$	$\mathcal{O}(t)$

Tabulka 2.2: Odhady složitostí těla hlavního cyklu pro různé typy fuzzy rozkladů.

Celkově tak pro složitost algoritmu DFT dostáváme odhady $\mathcal{O}(Nd2^d)$ při použití rovnoměrného kartézského fuzzy rozkladu, $\mathcal{O}(N\log(k))$ při obecném kartézském fuzzy rozkladu a $\mathcal{O}(Nt)$ v případě triangulárního fuzzy rozkladu.

Prozatím jsme na vstupní data $\langle X_i; y_i \rangle$ nekladli žádné další požadavky, především co se týče jejich uspořádání. V praxi se však často vyskytují úlohy, při kterých můžeme pořadí bodů X_i, ve kterých je hodnota funkce f známa, ovlivnit, respektive přímo sami zvolit.

Ideální situace nastává, když data splňují podmínku:

$$\mathrm{ref}(X_{i+1}) - \mathrm{ref}(X_i) \leq 1 \qquad \text{pro všechna } i = 1, \ldots, k-1, \tag{2.38}$$

prakticky však stačí i o něco slabší podmínka:

$$\mathrm{ref}(X_i) \leq \mathrm{ref}(X_{i+1}) \qquad \text{pro všechna } i = 1, \ldots, k-1, \tag{2.39}$$

což znamená, že jsou data uspořádána tak, že pořadí jim odpovídajících referenčních komponent koresponduje s pořadím výstupních komponent $F_1, \ldots, F_k$.

Pokud vstupní data splňují podmínku (2.38), resp. (2.39), můžeme v optimalizaci algoritmu DFT nahradit hledání referenční komponenty odpovídající bodu X_i kontrolou, zda je tato referenční komponenta stejná jako v předchozím kroku, jejíž index jsme si zapamatovali v pomocné proměnné.

Dokud tedy platí:

$$\mathrm{ref}(X_i) = \mathrm{ref}(X_{i-1}),$$

což zjistíme pomocí kontroly zda

$$A_{\mathrm{ref}(X_{i-1})}(X_i) > 0 \tag{2.40}$$

v čase $\mathcal{O}(a)$, nemusíme referenční komponentu měnit. Pokud vstupní data splňují podmínku (2.38), stačí navýšit hodnotu indexu referenční komponenty o 1. Pokud platí pouze podmínka (2.39) provádíme inkrementaci indexu referenční komponenty, dokud nenalezneme správnou hodnotu $\mathrm{ref}(X_i)$ splňující podmínku $A_{\mathrm{ref}(X_i)}(X_i) > 0$.

Ještě slabším požadavkem na vstupní data, který má stále dobré vlastnosti co se týče jeho příznivého vlivu na celkovou složitost algoritmu DFT, je seskupení vstupních dat do shluků podle referenčních komponent, resp. podle odpovídajících spádových oblastí fuzzy rozkladu:

$$\mathrm{ref}(X_i) \neq \mathrm{ref}(X_{i+1}) \Rightarrow (\forall j > i) : \ \mathrm{ref}(X_i) \neq \mathrm{ref}(X_j) \qquad \text{pro všechna } i = 1, \ldots, k-1, \tag{2.41}$$

což vede k tomu, že odpovídající spádovou oblast je nutno vyhledávat nejvýše k krát.

Za těchto uvedených podmínek (2.38), (2.39) a (2.41) se částečně stírají rozdíly způsobené odlišnou rychlostí nalezení odpovídajících komponent ovlivněných zpracovávanými daty způsobené různými vlastnostmi jednotlivých typů fuzzy rozkladů. Složitost hlavního cyklu je nyní dána součtem složitostí odpovídající kontrole, zda aktuální bod X_i spadá do označené oblasti, dále složitostí související se změnou a hledáním odpovídající referenční komponenty (resp. spádové oblasti v případě triangulárního fuzzy rozkladu) a konečně složitostí samotné aktualizace ovlivněných komponent. Odhady výše uvedených celkových počtů operací jsou shrnuty v tabulkách 2.3, 2.4 a 2.5.

Algoritmus inverzní F-transformace

Jak již bylo dříve zmíněno, inverzní F-transformaci používáme ve druhé fázi aproximace funkce f ve tvaru (2.24). Pro praktické využití potřebujeme algoritmus (IFT), který pro zadaný fuzzy rozklad $\mathcal{P} = \tilde{A}_1, \ldots, \tilde{A}_k$ a vektor komponent $\mathbf{F} = [F_1, \ldots, F_k]$ vypočítá funkční hodnotu inverzní F-transformace $f^{(-1)}_{\mathcal{P},\mathbf{F}}$ v libovolném bodě $X \in \mathcal{D}(f)$. Návrh tohoto algoritmu je ve výpisu Alg. 2.

$\mathrm{ref}(X_{i+1}) - \mathrm{ref}(X_i) \leq 1$	Kartézský fuzzy rozklad		Triangulární fuzzy rozklad
	rovnoměrný	obecný	
Kontrola $A(X) > 0$	$\mathcal{O}(Na)$	$\mathcal{O}(Na)$	$\mathcal{O}(Na)$
Změna spádové oblasti	$\mathcal{O}(d + kd2^d)$	$\mathcal{O}(\log(k) + kd2^d)$	$\mathcal{O}(t)$
Aktualizace komponent	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Nad)$
Celková složitost	$\mathcal{O}(Na2^d + kd2^d)$	$\mathcal{O}(\log(k) + Na2^d + kd2^d)$	$\mathcal{O}(t + Nad)$
při $a \in \mathcal{O}(d)$, $n \gg k$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd^2)$

Tabulka 2.3: Odhady celkového počtu operací za podmínky (2.38).

$\mathrm{ref}(X_i) \leq \mathrm{ref}(X_{i+1})$	Kartézský fuzzy rozklad		Triangulární fuzzy rozklad
	rovnoměrný	obecný	
Kontrola $A(X) > 0$	$\mathcal{O}((N + k)a)$	$\mathcal{O}((N + k)a)$	$\mathcal{O}((N + k)a)$
Změna spádové oblasti	$\mathcal{O}(d + kd2^d)$	$\mathcal{O}(\log(k) + kd2^d)$	$\mathcal{O}(t)$
Aktualizace komponent	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Nad)$
Celková složitost	$\mathcal{O}(Na2^d + kd2^d)$	$\mathcal{O}(\log(k) + Na2^d + kd2^d)$	$\mathcal{O}(t + Nad)$
při $a \in \mathcal{O}(d)$, $n \gg k$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd^2)$

Tabulka 2.4: Odhady celkového počtu operací za podmínky (2.39).

Algoritmus 2: Inverzní F-transformace

označení : IFT

vstup : $\{X_i\}_1^N$ – soubor N bodů $X_i \in \mathbb{R}^d$ X – bod z definičního oboru $\mathcal{D}(f) \subseteq \mathbb{R}^d$,
$\mathcal{P} = \{\tilde{A}_j\}_1^k$ – systém fuzzy množin – fuzzy rozklad definičního oboru $\mathcal{D}(f)$
$[F_1, \ldots, F_k]$ – k-tice reálných komponent

výstup : f – hodnota $f^{(-1)}_{\mathcal{P},\mathbf{F}}(X)$ – funkční hodnota inverzní F-transformace v bodě X

```
// inicializace
```
$f := 0$;

```
// aktualizace hodnoty výsledku pomocí všech komponent,
// v jejichž oblasti působnosti se nachází bod X
```
foreach j : $A_j(X) > 0$ **do**
  $f := f + F_j A_j(X)$;
end

return f;

Z praktického hlediska má širší uplatnění algoritmus hromadné inverzní F-transformace, jehož podoba je ve výpisu Alg. 3. Jedná se v podstatě o algoritmus IFT aplikovaný postupně na více

$\mathrm{ref}(X_i) \neq \mathrm{ref}(X_{i+1}) \Rightarrow$ $(\forall j > i):\ \mathrm{ref}(X_i) \neq \mathrm{ref}(X_j)$	Kartézský fuzzy rozklad		Triangulární fuzzy rozklad
	rovnoměrný	obecný	
Kontrola $A(X) > 0$	$\mathcal{O}(Na)$	$\mathcal{O}(Na)$	$\mathcal{O}(Na)$
Změna spádové oblasti	$\mathcal{O}(kd + kd2^d)$	$\mathcal{O}(k\log(k) + kd2^d)$	$\mathcal{O}(kt)$
Aktualizace komponent	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Na2^d)$	$\mathcal{O}(Nad)$
Celková složitost	$\mathcal{O}(Na2^d + kd2^d)$	$\mathcal{O}(k\log(k) + Na2^d + kd2^d)$	$\mathcal{O}(t + Nad)$
při $a \in \mathcal{O}(d)$, $n \gg k$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd^2)$

Tabulka 2.5: Odhady celkového počtu operací za podmínky (2.41).

vstupních dat X_i. Z výpisu tohoto algoritmu je hned na první pohled patrné, že má z hlediska výpočetní složitosti stejnou strukturu jako algoritmus DFT a lze tudíž na jeho analýzu uplatnit stejné myšlenkové postupy. V tabulce 2.6 jsou obsaženy přehledy výsledných odhadů složitosti obou algoritmů DFT a IFT_N.

Algoritmus 3: Hromadná inverzní F-transformace

označení : IFT_N

vstup : $\{X_i\}_1^N$ – soubor N bodů $X_i \in \mathcal{D}(f) \subseteq \mathbb{R}^d$,
$\mathcal{P} = \{\tilde{A}_j\}_1^k$ – systém fuzzy množin – fuzzy rozklad definičního oboru $\mathcal{D}(f)$,
$[F_1, \ldots, F_k]$ – k-tice reálných komponent

výstup : $\{y_i\}_1^N$ – výstupní soubor funkčních hodnot inverzní F-transformace
$y_i = f_{\mathcal{P},\mathbf{F}}^{(-1)}(X_i)$

```
// hlavní cyklus pro všechna vstupní data X_i
for i := 1 to N do
    y_i := 0;
    // aktualizace hodnoty y_i pomocí všech komponent,
    // v jejichž oblasti působnosti se nachází bod X_i
    foreach j : A_j(X_i) > 0 do
        y_i := y_i + F_j A_j(X);
    end
end
return {y_i}_1^N;
```

2.4.1 Shrnutí a srovnání

- Z uvedených výsledků výpočetní složitosti je patrné, že už při splnění podmínky (2.41), tedy když vstupní data jsou seskupena ve shlucích podle spádové oblasti, se stírá rozdíl při použití rovnoměrného či obecného kartézského rozkladu. Tohoto faktu se používá ve většině praktických aplikací souvisejících s aproximací známé funkce f.

Požadavky kladené na vstupní data	Kartézský fuzzy rozklad		Triangulární fuzzy rozklad
	rovnoměrný	obecný	
Bez omezení	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(N\log(k))$	$\mathcal{O}(Nt)$
Alespoň podmínka (2.41)	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd2^d)$	$\mathcal{O}(Nd^2)$

Tabulka 2.6: Odhady složitosti algoritmů DFT a IFT_N za předpokladu $a \in \mathcal{O}(d)$, $N \gg k$.

- Rozdíl ve výpočetní složitosti mezi kartézskou a triangulární F-transformací je dán tvarem (zejména počtem vrcholů) jednotlivých segmentů, ze kterých je sestaven fuzzy rozklad. Zatímco v případě kartézské F-transformace se jedná o d-rozměrné kvádry o 2^d vrcholech, u triangulární F-transformace jsou to d-rozměrné simplexy sestavené z $d+1$ nadrovin s celkovým počtem $d+1$ vrcholů.
- S použitím triangulární F-transformace je spojena jedna nevýhoda spočívající ve vyšších paměťových nárocích spojených s reprezentací datové struktury popisující triangulární fuzzy rozklad.
- Triangulární F-transformace je vhodná především pro řešení úloh spočívajících na metodě konečných prvků, kdy už máme provedenu triangularizaci zkoumané oblasti.
- Triangulární F-transformace je výhodné použít v případech, kdy potřebujeme ve fuzzy rozkladu provést *lokální* zjemnění a požadujeme, aby zbytek fuzzy rozkladu zůstal nezměněn. V případě kartézského fuzzy rozkladu toho nemůžeme dosáhnout, protože sebemenší změna i pouze v jedné (i-té) dimenzi, spočívající v přidání jednoho uzlu, způsobí přidání k/k_i komponent a ovlivní hodnoty celého d-rozměrného pásu obsahujícího $2k/k_i$ komponent.

2.5 Optimalizace chyby fuzzy transformace

V této části se budeme krátce věnovat aproximační chybě při použití F-transformace a faktorům, které mohou ovlivnit její velikost. V dalším budeme pracovat s chybou aproximace určenou jako maximální absolutní odchylku od původní funkce f.

Definice 23 (Chyba aproximace).
Nechť je dána funkce f a její aproximace f^A na intervalu $[a;b]$. Chybu aproximace definujeme jako

$$\varepsilon = \max_{x\in[a;b]} |f(x) - f^A(x)|. \tag{2.42}$$

Pro další úvahy je nezbytná následující věta, jejíž platnost nám zaručuje možnost snižování chyby aproximace funkce f pomocí F-transformace na libovolně nízkou hodnotu.

Věta 3.
Nechť f je reálná spojitá funkce na $[a;b]$. Pak ke každému $\varepsilon > 0$ existuje n_ϵ a rovnoměrný fuzzy rozklad $\mathcal{P} = \{\tilde{A}_i, \ldots, \tilde{A}_{n_\varepsilon}\}$ intervalu $[a;b]$ takový, že pro všechna $x \in [a;b]$:

$$|f(x) - f^{(-1)}_{\mathcal{P},f}(x)| \leq \varepsilon,$$

kde $f^{(-1)}_{\mathcal{P},f}(x)$ je inverzní F-transformace funkce f vzhledem k fuzzy rozkladu $\mathcal{P}$ definovaná v (2.24).

Díky platnosti této věty, jejíž důkaz je uveden v [25] jako theorem 2, můžeme prakticky neomezeně zvyšovat přesnost aproximace funkce f určené prostřednictvím inverzní F-transformace. Velikost chyby aproximace je dána jednak charakterem aproximované funkce f, ale co je pro nás podstatnější, tak závisí především na volbě fuzzy rozkladu intervalu $[a; b]$. Z hlediska velikosti chyby aproximace mají na její hodnotu vliv následující vlastnosti fuzzy rozkladu:

- počet bázických funkcí
- tvar bázických funkcí
- rozložení uzlů fuzzy rozkladu v rámci intervalu $[a; b]$

Vzhledem k tomu, že v praxi se tvar bázických funkcí prokázal jako nejméně významný parametr, budeme se dále zabývat pouze počtem a rozložením uzlů, kterými je dán fuzzy rozklad intervalu $[a; b]$ při pevně zvoleném tvaru bázických funkcí – nejčastěji trojúhelníkovém.

Pokud bychom uvažovali pouze rovnoměrný rozklad intervalu $[a; b]$, je rozložení vnitřních uzlů v rámci tohoto intervalu pevně dáno parametrem počtu bázických funkcí n. V této souvislosti má smysl úloha nalezení takového minimálního n, pro které je chyba aproximace funkce f menší než předem zadaná hodnota ε.

Řešení této úlohy pro diskrétní n-rozměrné funkce zadané tabulkou hodnot bylo implementováno do systému LFLC 2000. Algoritmus hledání minimálního počtu bázických funkcí je založen na principu půlení intervalu, kdy se od počáteční dostatečně vysoké hodnoty n_0 postupně vypočítávají aproximace s nižším počtem bázických funkcí podle pravidel metody půlení počátečního intervalu $[0; n_0]$.

Úloha minimalizace chyby aproximace se stává mnohem zajímavější, když uvažujeme obecný fuzzy rozklad intervalu $[a; b]$. V tomto případě se již jedná o klasickou optimalizační úlohu se všemi jejími náležitostmi. Z toho vyplývá samozřejmě celá řada přístupů, jak danou úlohu řešit. Jednou z možností je pohlížet na zpětnou F-transformaci jako na dopřednou radiální neuronovou síť [40], kde komponenty F-transformace tvoří váhy spojů směřujících z vnitřní skryté vrstvy do vrstvy výstupní.

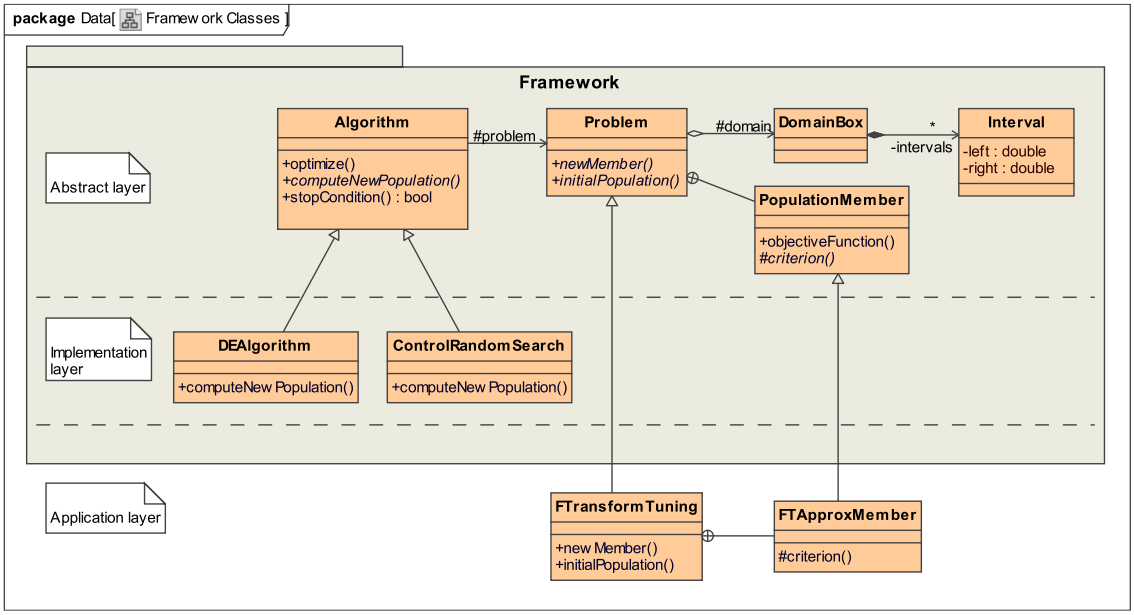
Dalším možným přístupem je využití evolučních algoritmů [9]. Při řešení byly využity algoritmy implementované v rámci vyvíjené softwarové knihovny `IRAFMlib`. Součástí této knihovny je implementace evolučních algoritmů sloužících pro řešení obecných optimalizačních úloh [6, 7]. Z této knihovny byly pro minimalizaci chyby aproximace využity algoritmy založené na diferenciální evoluci [41, 42, 43].

Prakticky řešení spočívá v doimplementování aplikační vrstvy do obecného rozhraní, které knihovna `IRAFMlib` nabízí – viz obr. 2.14. Aplikační vrstva obsahuje dvě třídy implementující specifiké rysy dané úlohy, na jejíž řešení jsou evoluční algoritmy knihovny použity. V našem případě se jedná o třídy:

– *FTransformTuning* specifikuje nastavení parametrů optimalizační úlohy v podobě aproximované funkce f společně s intervalem $[a; b]$ a zajišťuje především vytvoření počáteční populace.

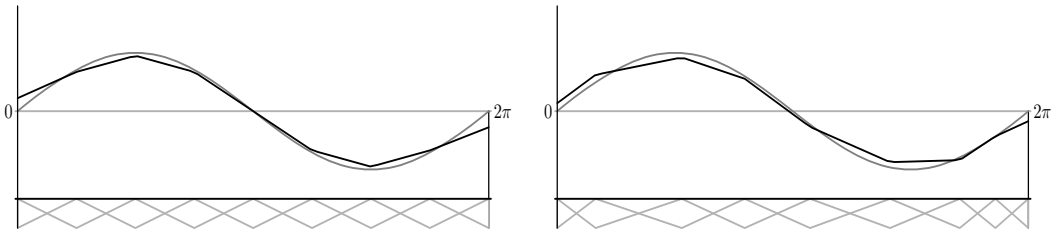
– *FTApproxMember* specifikuje vlastnosti člena populace, což v našem případě představuje nastavení F-transformace spočívající v rozložení uzlů tvořících fuzzy rozklad intervalu $[a; b]$. Hlavní úlohou této třídy je implementování metody *criterion()*, která v našem případě

vypočítává chybu aproximace při použití F-transformace s odpovídajícím rozložením uzlů fuzzy rozkladu.



Obrázek 2.14: Rozhraní knihovny `IRAFMlib` pro použití evolučních algoritmů.

Během optimalizačního procesu se datové hodnoty členů populace, v podobě n-rozměrného vektoru představujícího rozmístění uzlů rozkladu, vyvíjí dle pravidel diferenciální evoluce, dokud není splněno optimalizační kritérium dosažením požadované přesnosti, anebo překročen maximální počet iterací.



Obrázek 2.15: Porovnání výsledků – vlevo rovnoměrný rozklad, vpravo po optimalizaci.

Pro ilustraci je na obrázku 2.15 ukázáno srovnání dvou příkladů aproximace funkce $sin(x)$ na intervalu $[0; 2\pi]$. Fuzzy rozklad má v obou případech 7 vnitřních uzlů. V prvním případě byl použit rovnoměrný rozklad pro srovnání s výsledkem po optimalizaci. Aproximační chyba při rovnoměrném rozkladu je 0.234, při optimalizovaném rozložení uzlů se zlepšila na 0.136.

2.6 Zpracování signálu – fuzzy filtr

Fuzzy transformace je díky svým vlastnostem aplikovatelná i na oblast zpracování signálu. Zde je možno ji použít pro filtrování jednorozměrného signálu nebo také v dvojrozměrném případě pro

zpracování obrazu.

Z hlediska zpracování signálu má F-transformace pro použití jako filtru dvě hlavní uplatnění:

1. Odstranění šumu ze signálu
2. Vyhlazení průběhu signálu

Na tomto místě je nutné zdůraznit, že obsah této sekce je nutno chápat jako podklad pro další rozpracování a výzkum použitelnosti fuzzy metod při zpracování signálu. V současnosti se pracuje na teoretickém odhadu vlivu F-transformace na rozptyl náhodné veličiny a do budoucna se uvažuje o bližší analýze frekvenční charakteristiky fuzzy filtrů.

Ve skutečnosti nebudeme uvažovat šum v technickém slova smyslu jakožto náhodnou funkci s určitými charakteristikami, ale jen jako nenáhodnou funkci poruchy. Přesto budeme dále používat termín *šum*.

2.6.1 Odstranění šumu – inverzní fuzzy filtr

Pro použití F-transformace na odstranění šumu se využívá důležitého rysu F-transformace, který je dán tím, že má vlastnosti lineárního operátoru, což můžeme vyjádřit následující větou, jejíž platnost je ukázána v [25] jako lemma 3.

Věta 4.
Mějme dány dvě spojité funkce f, g na intervalu $[a;b]$ a fuzzy rozklad $\mathcal{P}$ intervalu $[a;b]$. Potom pro libovolné $\alpha, \beta \in \mathbb{R}$ platí:

$$\mathbf{F}_{\mathcal{P}}[\alpha f + \beta g] = \alpha \mathbf{F}_{\mathcal{P}}[f] + \beta \mathbf{F}_{\mathcal{P}}[g].$$

Díky této vlastnosti můžeme F-transformaci použít za jistých podmínek [26, 44] pro filtrování šumu. Myšlenka je založena na tom, že pokud je k funkci f přidán *aditivní* šum, jehož F-transformace má nulové komponenty a tudíž jeho inverzní F-transformace je rovna nulové funkci, bude inverzní F-transformace tohoto součtu rovna inverzní F-transformaci původní funkce f nezatížené šumem. Následující věty z [44] (theorem 3 a 4) udávají podmínky kladené na šum, aby byl *odstranitelný*.

Věta 5.
Nechť $\tilde{A}_1, \ldots, \tilde{A}_n$ je rovnoměrný fuzzy rozklad intervalu $[a;b]$ takový, že $h = \dfrac{b-a}{n-1}$ a $n > 2$. Dále mějme dánu spojitou, periodickou funkci $e(x)$ s periodou $2h$, pro kterou navíc platí:

$$e(x_k - x) = -e(x_k + x) \qquad \text{na intervalu } [x_{k-1}; x_{k+1}],$$

kde $k = 2, \ldots, n-1$ a $x \leq h$. Potom regulární *komponenty přímé F-transformace funkce $e(x)$ vzhledem k $\tilde{A}_1, \ldots, \tilde{A}_n$ jsou rovny 0, tedy*

$$\int_{x_{k-1}}^{x_{k+1}} A_k(x) e(x) \,\mathrm{d}x = 0 \qquad \text{pro } k = 2, \ldots, n-1.$$

Věta 6.
Nechť $\tilde{A}_1, \ldots, \tilde{A}_n$ je rovnoměrný fuzzy rozklad intervalu $[a;b]$ takový, že $h = \dfrac{b-a}{n-1}$ a $n > 2$. Dále mějme dánu spojitou funkci $e(x)$ definovanou na intervalu $[a;b]$, pro kterou platí:

$$e(x + h) = e(x) \qquad \text{pro } x \in [a; b-h]$$

a

$$\int_{x_{k-1}}^{x_k} e(x)\,\mathrm{d}x = 0.$$

Potom regulární *komponenty přímé F-transformace funkce* $e(x)$ *vzhledem k fuzzy rozkladu* $\tilde{A}_1, \ldots, \tilde{A}_n$ *jsou rovny 0, tedy*

$$\int_{x_{k-1}}^{x_{k+1}} A_k(x)e(x)\,\mathrm{d}x = 0 \qquad \textit{pro } k = 2, \ldots, n-1.$$

Pokud tedy funkce $e(x)$ představující aditivní šum k funkci $f(x)$ splňuje výše uvedené podmínky na odstranitelnost vzhledem k fuzzy rozkladu $\mathcal{P}$, můžeme podle notace (2.24) psát:

$$f^{(-1)}_{\mathcal{P},f+e}(x) = f^{(-1)}_{\mathcal{P},f}(x) \tag{2.43}$$

Platnost vztahu (2.43) je matematickým pozadím, na jehož základě jsou postaveny všechny algoritmy používající inverzní F-transformaci pro odstranění aditivního šumu. Tato rovnost ilustruje možnosti F-transformace při odstranění poruchové funkce. Pokud by skutečný náhodný šum byl „dostatečně pravidelný“, lze předpokládat, že jeho podstatná část bude odstraněna. Dosud prováděné experimenty tento předpoklad potvrzují.

Pokud máme hodnoty funkce f, u které předpokládáme aditivní šum splňující podmínky odstranitelnosti, přímo dány v celém jejím definičním oboru, můžeme pro odstranění tohoto šumu použít přímo dříve uvedené algoritmy alg. 1 a alg. 3 pro dopřednou a hromadnou zpětnou F-transformaci. Jiná je však situace při zpracování signálu, kdy většinou ani nevíme kolik hodnot signálu budeme zpracovávat. Ještě důležitější však při filtrování signálu bývá požadavek, aby filtr pracoval v tzv. *online* režimu. Tedy, aby filtrované hodnoty byly přístupné nejlépe okamžitě po přijmutí originálních hodnot signálu, resp. maximálně s předem daným časovým zpožděním (*time delay*).

Vstupní signál

Za vstupní signál budeme dále považovat časovou funkci $w(t)$, jejíž přicházející hodnoty zaznamenáváme předem danou vzorkovací frekvencí $f = 1/T$. Pracujeme tedy s diskrétní posloupností hodnot:

$$w(0),\ w(T),\ w(2T),\ \ldots,\ w(iT),\ \ldots,$$

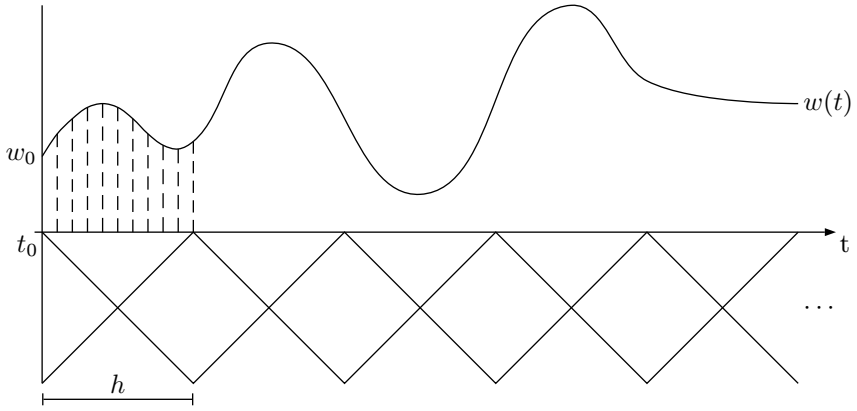
kterou můžeme přepsat do podoby

$$\{w(iT)\}_{i=0}^{\infty} = w_0,\ w_1,\ \ldots,$$

přičemž v čase $t_i = iT$ je naměřena hodnota w_i, takže do té doby máme informace pouze o hodnotách $w_0, \ldots, w_i$.

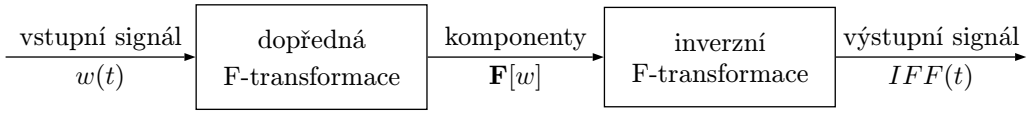
Vzhledem k tomu, že při zpracování signálu jsme často v situaci, kdy neznáme funkční hodnoty časové funkce $w(t)$ všechny najednou, ale přicházejí postupně s narůstajícím časem, musíme tomu přizpůsobit i algoritmus výpočtu komponent F-transformace. Kvůli požadavku na rychlost odezvy jsme nuceni vypočítávat hodnoty inverzní F-transformace jak jen to bude možné a rovněž tomu přizpůsobit odpovídajícím způsobem algoritmus výpočtu.

Dostáváme se tak k upravenému algoritmu, který vypočítává průběžně hodnoty komponent F-transformace funkce $w(t)$ a po jejich vyčíslení vzápětí vrací funkční hodnoty inverzní F-transformace jako výstup představující filtrovaný výstupní signál. Celkově tímto sekvenčním spojením



Obrázek 2.16: Fuzzy rozklad časové osy pro filtraci signálu. Pod časovou osou t je zobrazeno rozmístění bázických funkcí.

algoritmů dopředné a zpětné F-transformace získáme *inverzní fuzzy filtr* (viz obr. 2.17), který původní vstupní signál $w(t)$ převádí na výstupní filtrovaný signál $IFF(t)$.



Obrázek 2.17: Blokové schéma inverzního fuzzy filtru.

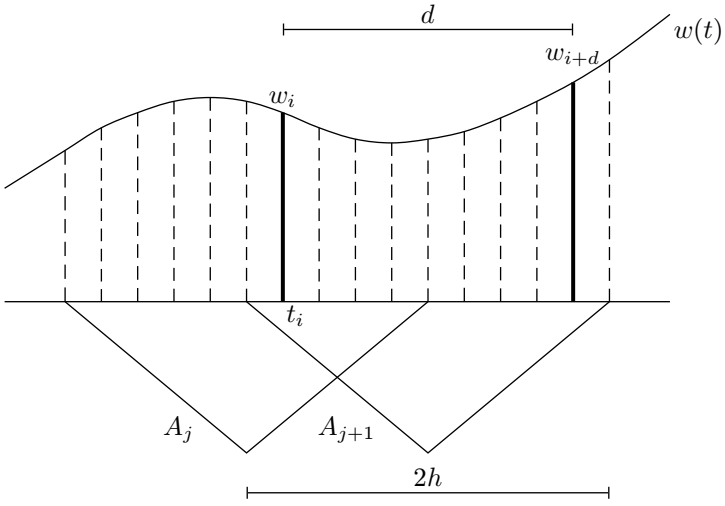
Jak je z podmínek uvedených ve větách 5 a 6 patrné, hraje pro úspěšné odstranění šumu ze signálu klíčovou roli parametr h (viz obr. 2.16) udávající počet vzorků signálu, které jsou zaznamenány v rámci jednoho úseku fuzzy rozkladu. Z toho je zřejmé, že v našem případě bude parametr h nabývat pouze celočíselných hodnot. Hodnota $2h$ pak představuje šířku základny bázické funkce fuzzy rozkladu obsahující $2h-1$ naměřených hodnot[7], z čehož vyplývá rozumný požadavek, aby $h \geq 2$.

Obrázek 2.18 znázorňuje původ časového zpoždění fuzzy filtru. Na obrázku je vyznačen časový okamžik t_i představující časový mezník, ve kterém je nutno znát hodnotu další komponenty. Pro výpočet hodnoty $IFF(t_i)$ filtrovaného signálu odpovídajícího tomuto okamžiku je nutno znát hodnotu komponenty příslušející bázické funkci A_{j+1}, kde $i = jh+1$. Pro výpočet hodnoty komponenty F_{j+1} však potřebujeme znát dalších $d = 2h-2$ hodnot původního signálu w. Z toho tedy dostáváme výsledné časové zpoždění dT fuzzy filtru, kdy až s příchodem hodnoty w_{i+d} v čase t_{i+d} jsme schopni vypočíst hodnotu výstupního signálu odpovídajícího hodnotě w_i.

Než přistoupíme k určení výpočetní složitosti výpočtu hodnot výstupního signálu z fuzzy filtru, definujme nejprve co vlastně budeme považovat za elementární operace, jejichž počet bude rozhodující pro určení konkrétní výpočetní složitosti daných algoritmů.

Definice 24 (Elementární operace).

[7] krajní hodnoty mají nulový stupeň příslušnosti



Obrázek 2.18: Časové zpoždění fuzzy filtru.

Mezi elementární operace budeme řadit základní aritmetické operace mezi číselnými hodnotami vyskytující se v matematických výrazech, tedy: $+, -, *, /, \mathrm{mod}, \mathrm{div}$.

Složitost matematických výrazů vzniklých složením konečného počtu elementárních operací je dána celkovým počtem základních aritmetických operací nutných k jejich vyhodnocení. Při určování konkrétní složitosti algoritmů budeme zanedbávat přiřazování hodnot proměnným a operace související s režií řídících proměnných cyklů. Například vyhodnocení výrazu $\sum_{i=1}^{n} a_i b_i$ spočívajícího v sečtení hodnot $a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ představuje n operací násobení a $n-1$ operací sčítání, tedy dohromady $2n-1$ elementárních aritmetických operací.

Výpočetní složitost inverzního fuzzy filtru

Co se týče výpočetní složitosti filtrace signálu metodou inverzního fuzzy filtru, bude zřejmě pro výpočet h filtrovaných hodnot zapotřebí $\mathcal{O}(h)$ operací. Vzhledem k charakteru úlohy, probíhající nejčastěji v reálném čase, kdy může být každá operace navíc kritická, je vhodné pokusit se určit počet nutných operací přesněji.

Věta 7.
Pro výpočet jedné filtrované hodnoty pomocí inverzního fuzzy filtru potřebujeme nejvýše 7 elementárních aritmetických operací.

Důkaz 2. Počet operací nutných pro určení hodnoty komponenty F_{j+1}, jejíž hodnotu využijeme pro výpočet h následujících hodnot odpovídajících časovým okamžikům $t_i, \ldots, t_{i+h-1}$, zjistíme ze vztahu pro její výpočet:

$$F_{j+1} = \frac{\sum_{k=i}^{i+d} A_{j+1}(t_k) w(t_k)}{h} = \frac{\sum_{k=i}^{i+d} A_{j+1}(t_k) w_k}{h},$$

z čehož dostáváme $d+1$ násobení, přičemž jedno z nich představuje násobení hodnotou 1, takže počítáme d násobení, d sčítání a jedno dělení, tedy celkem $2d+1 = 4h-3$ aritmetických operací

vzhledem k tomu, že $d = 2h - 2$, přičemž předpokládáme, že funkční hodnoty bázické funkce A_j máme předem spočítány ve všech bodech a uloženy v poli s přímým přístupem.

Když známe hodnotu komponenty F_{j+1}, můžeme spočítat $h-1$ následujících hodnot výsledného signálu podle vztahu inverzní F-transformace:

$$IFF(t_k) = F_j A_j(t_k) + F_{j+1} A_{j+1}(t_k) \qquad \text{pro } k = i, \ldots, i + h - 2,$$

přičemž h-tá následující hodnota je rovna přímo komponentě F_{j+1}. Pro výpočet každé hodnoty tedy potřebujeme dvě násobení a jedno sčítání, což nám dává $3(h-1)$ operací.

Celkově tedy pro výpočet h hodnot výstupního signálu potřebujeme $7h-6$ operací. Z toho pro výpočet jedné hodnoty dostáváme horní odhad $7 - \frac{6}{h} < 7$. Tím jsme ukázali, že výpočetní nároky na výpočet jedné filtrované hodnoty výstupního signálu nepřesahují 7 elmentárních aritmetických operací.

□

2.6.2 Filtr pro vyhlazení průběhu

Fuzzy filtr uvedený v předchozí části prokazuje dobré vlastnosti také při použití na vyhlazení průběhu signálu, resp. obecné funkce f. Pro vyhlazení průběhu můžeme ale využít také modifikované verze fuzzy filtrů popsaných dále.

Klouzavý fuzzy filtr

Hlavní myšlenka spočívá v odlišném chování fuzzy rozkladu filtru vzhledem k signálu. V předchozí části byla časová funkce vstupního signálu $w(t)$ pevně spojená s fuzzy rozkladem $\tilde{A}_0, \tilde{A}_1, \ldots$ časové osy t. Nyní budeme pracovat s představou, kdy vlna vstupního signálu $w(t)$ postupuje po časové ose s pevně zafixovaným fuzzy rozkladem. Dochází tak tedy k relativnímu posunu funkce $w(t)$ vůči fuzzy rozkladu časové osy t. Dostáváme tedy v podstatě posloupnost funkcí $w^0(t), w^1(t), \ldots$ pro jednotlivé časové okamžiky $t_0, t_1, \ldots$, přičemž platí:

$$w^i(t) = w(t + t_i) = w(t + iT) \qquad \text{pro všechna } i \geq 0.$$

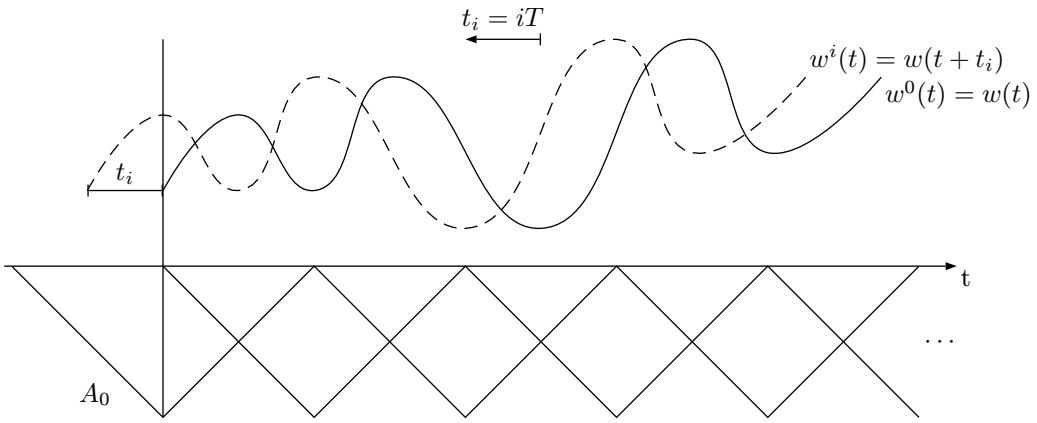
Úloha výpočtu komponent je tak vlastně převedena na postupnou aktualizaci hodnoty jediné komponenty F_0 příslušející bázické funkci A_0, která má vrchol v bodě $t = 0$. Na obrázku 2.20 je zobrazena situace v čase t_i, ve kterém se pracuje s funkcí $w^i(t)$, přičemž počátkem aktuálně prochází hodnota w_i. Na obrázku je také vyznačeno časové zpoždění $d = h - 1$ tohoto fuzzy filtru dané potřebou znalosti hodnot $w_{i-d}, \ldots, w_{i+d}$ pro výpočet aktuální hodnoty komponenty F_0 v čase t_i, kterou značíme F_0^i.

Hodnoty výstupního signálu odpovídající časovému okamžiku t_i jsou pak rovny přímo hodnotě komponenty F_0^i.

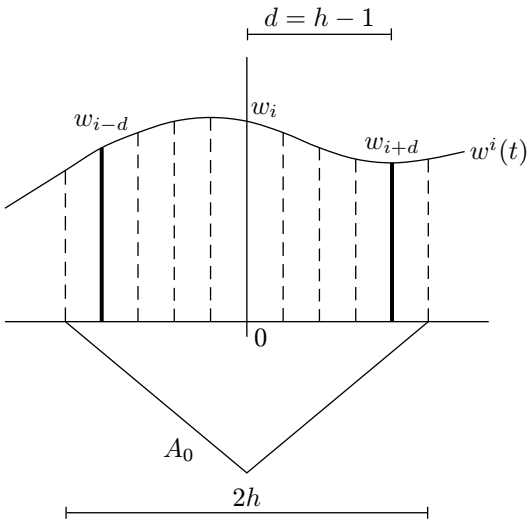
$$KFF(t_i) = F_0^i,$$

jejíž hodnota je vypočítána obdobně jako v předešlém případě podle vzorce:

$$F_0^i = \frac{\sum_{k=-d}^{d} A_0(t_k) w^i(t_k)}{h} = \frac{\sum_{k=-d}^{d} A_0(t_k) w(t_{i+k})}{h} = \frac{\sum_{k=1-h}^{h-1} A_0(t_k) w_{i+k}}{h}. \tag{2.44}$$



Obrázek 2.19: Relativní posun signálu $w(t)$ vůči fuzzy rozkladu časové osy t.



Obrázek 2.20: Umístění bázické funkce A_0 odpovídající průběžně aktualizované komponentě F_0.

Pokud pro nás není rozhodující rychlost výpočtu, můžeme hodnoty výstupního signálu vypočítávat přímo podle vztahu (2.44), který lze přímočaře a přehledně implementovat do odpovídajícího algoritmu *klouzavého fuzzy filtru*. Pro výpočet každé hodnoty výstupního signálu je v tomto případě zapotřebí $2h - 3$ násobení, $2h - 3$ sčítání a jedno dělení, tedy $4h - 5$ elementárních aritmetických operací.

Optimalizace výpočtu klouzavého fuzzy filtru

Pokud jsme však v situaci, kdy je rychlost výpočtu důležitá, můžeme se pokusit o optimalizaci tohoto přímočarého algoritmu. Nejprve si všimněme faktu vyplývajícího z podmínek kladených na fuzzy rozklad intervalu uvedených v definici 10. Pro hodnoty bázické funkce A_k na intervalu $[x_{k-1}; x_{k+1}] = [x_k - h; x_k + h]$ rovnoměrného fuzzy rozkladu obecně platí:

$$A_k(x_k - h + x) = 1 - A_k(x_k + x) \qquad \text{pro } 0 \leq x \leq h,$$

což přepsáno do právě používané symboliky pro bázickou funkci A_0 znamená:

$$A_0(t_{k-h}) = 1 - A_0(t_k) \qquad \text{pro } k = 0, \ldots, h. \tag{2.45}$$

Toho můžeme využít pro výpočet části sumy ze vzorce (2.44) pro výpočet hodnoty komponenty. Nejprve rozdělme zmíněnou sumu na tři části:

$$\sum_{k=1-h}^{h-1} A_0(t_k) w_{i+k} = \sum_{k=1-h}^{-1} A_0(t_k) w_{i+k} + w_i + \sum_{k=1}^{h-1} A_0(t_k) w_{i+k}. \tag{2.46}$$

Při označení

$$S_L^i = \sum_{k=1-h}^{-1} A_0(t_k) w_{i+k}$$

a

$$S_R^i = \sum_{k=1}^{h-1} A_0(t_k) w_{i+k} \tag{2.47}$$

dostaneme:

$$\sum_{k=1-h}^{h-1} A_0(t_k) w_{i+k} = S_L^i + w_i + S_R^i.$$

Zaměřme se na levou část výrazu v čase t_{i+h}:

$$S_L^{i+h} = \sum_{k=1-h}^{-1} A_0(t_k) w_{i+h+k}.$$

Po přeindexování dostaneme:

$$S_L^{i+h} = \sum_{k=1}^{h-1} A_0(t_{k-h}) w_{i+k}.$$

Nyní můžeme použít vztah (2.45) na hodnotu bázické funkce uvnitř sumy:

$$S_L^{i+h} = \sum_{k=1}^{h-1} (1 - A_0(t_k)) w_{i+k},$$

což můžeme přepsat do tvaru:

$$S_L^{i+h} = \sum_{k=1}^{h-1} w_{i+k} - \sum_{k=1}^{h-1} A_0(t_k) w_{i+k},$$

kde pravá suma představuje hodnotu S_R^i. Nakonec tedy máme:

$$S_L^{i+h} = \sum_{k=1}^{h-1} w_{i+k} - S_R^i. \tag{2.48}$$

Nyní je již patrná optimalizace spočívající ve využití jednou spočítané hodnoty S_R^i pro časový okamžik t_i, podruhé v čase t_{i+h} pro výpočet hodnoty S_L^{i+h}. Co se týče optimální implementace tohoto postupu, musíme v paměti udržovat h hodnot $S_L^i, \ldots, S_L^{i+h-1}$.

Nejlépe pro tyto účely poslouží datová struktura v podobě kruhové fronty reprezentované statickým polem o velikosti h. Časové nároky spojené se současným vložením prvku na konec fronty a vrácením prvku ze začátku fronty uvažujeme pouze v souvislosti s aktualizací indexu

ukazujícího na aktuální začátek fronty v rámci datového pole. Aktualizace spočívá v inkrementaci tohoto indexu a kontrole, zda nedošlo k překročení hranice pole dané hodnotou h. To lze provést například pomocí výrazu:

$$i := (i + 1) \bmod h,$$

obsahujícího dvě elementární aritmetické operace. Při extrémních nárocích na rychlost je možno následující hodnoty indexů předpočítat a uložit do pole, takže dále nebudeme složitost operací souvisejících s vkládáním a vyjímáním prvků započítávat.

Pro výpočet hodnoty S_L^{i+h}, kterou budeme do fronty ukládat v čase t_i, musíme znát ještě hodnotu sumy $S_w^i = \sum_{k=1}^{h-1} w_{i+k}$. Abychom nemuseli tento součet v každém časovém okamžiku t_i počítat vždy celý znovu, můžeme jeho hodnotu průběžně aktualizovat pomocí vztahu:

$$S_w^i = S_w^{i-1} + w_{i+h-1} - w_i.$$

Pro tuto možnost musíme v další kruhové frontě udržovat hodnoty $w_i, \ldots, w_{i+h-2}$ pro jejich pozdější použití.

Nyní již máme vše připraveno pro důkaz následujícího tvrzení:

Věta 8.
Nechť je dán právě uvedený klouzavý filtr s parametrem h, kde hodnota $2h-1$ představuje počet vzorků pokrytých základnou jeho bázické funkce. Celkový počet elementárních aritmetických operací nutných pro výpočet jedné filtrované hodnoty můžeme shora omezit hodnotou $2(h+1)$.

Důkaz 3. V každém časovém okamžiku t_i musíme vypočítat hodnotu součtu S_R^i (2.47), kterou využijeme jednak pro výpočet hodnoty komponenty v aktuálním čase pomocí vztahu (2.46) a zároveň ji použijeme pro výpočet hodnoty S_L^{i+h} podle vztahu (2.48), kterou uložíme pro pozdější využití. Celkově je tedy počet operací:

$= 2h-3$ operací pro výpočet $S_R^i = \sum_{k=1}^{h-1} A_0(t_k) w_{i+k}$

$+$ 2 operace pro výpočet $S_w^i = \sum_{k=1}^{h-1} w_{i+k} = S_w^{i-1} + w_{i+h-1} - w_i$

$+$ 2 operace pro výpočet $G^i = \sum_{k=1-h}^{h-1} A_0(t_k) w_{i+k} = S_L^i + w_i + S_R^i$

$+$ jedna operace dělení pro určení konečné hodnoty $KFF(t_i) = F_0^i = \dfrac{G^i}{h}$,

což dohromady dává 2(h+1) aritmetických operací, tedy v podstatě poloviční počet oproti původnímu algoritmu.

□

Trojúhelníkový fuzzy filtr

Výsledek předchozí optimalizace můžeme ještě zlepšit, pokud se omezíme pouze na trojúhelníkový tvar bázických funkcí rozkladu. V takovémto případě můžeme pro výpočet hodnoty S_R^i díky lineárnímu průběhu funkce A_0 využít předchozí hodnoty S_R^{i-1}.

Věta 9.
Nechť je dán klouzavý filtr dle předchozího popisu. Pokud je při jeho konstrukci použita bázická funkce trojúhelníkového tvaru, je pro výpočet výsledné filtrované hodnoty zapotřebí nejvýše 9 elementárních aritmetických operací.

Důkaz 4. Když je tvar bázické funkce A_0 trojúhelníkový, můžeme hodnoty její pravé části vyjádřit přímo jako:

$$A_0(t_k) = 1 - \frac{k}{h} \qquad \text{pro } k = 0, \ldots, h, \tag{2.49}$$

což po dosazení do (2.47) dává:

$$S_R^i = \sum_{k=1}^{h-1} A_0(t_k) w_{i+k} = \sum_{k=1}^{h-1} \left(1 - \frac{k}{h}\right) w_{i+k}.$$

Podobně můžeme vyjádřit hodnotu S_R^{i-1}:

$$S_R^{i-1} = \sum_{k=1}^{h-1} A_0(t_k) w_{i-1+k} = \sum_{k=1}^{h-1} \left(1 - \frac{k}{h}\right) w_{i-1+k}.$$

Po přeindexování dostaneme:

$$S_R^{i-1} = \sum_{k=0}^{h-2} \left(1 - \frac{k+1}{h}\right) w_{i+k},$$

což můžeme přepsat do podoby:

$$S_R^{i-1} = \sum_{k=0}^{h-2} \left(1 - \frac{k}{h}\right) w_{i+k} - \frac{1}{h} \sum_{k=0}^{h-2} w_{i+k}.$$

Nyní již můžeme vyjádřit S_R^i pomocí S_R^{i-1} následovně:

$$S_R^i = S_R^{i-1} - w_i + \left(1 - \frac{h-1}{h}\right) w_{i+h-1} + \frac{1}{h} \sum_{k=0}^{h-2} w_{i+k},$$

což můžeme přepsat na tvar:

$$S_R^i = S_R^{i-1} - w_i + \frac{1}{h} \sum_{k=0}^{h-1} w_{i+k},$$

který je výhodnější přepsat do podoby:

$$S_R^i = S_R^{i-1} - w_i + \frac{w_i + \sum_{k=1}^{h-1} w_{i+k}}{h}.$$

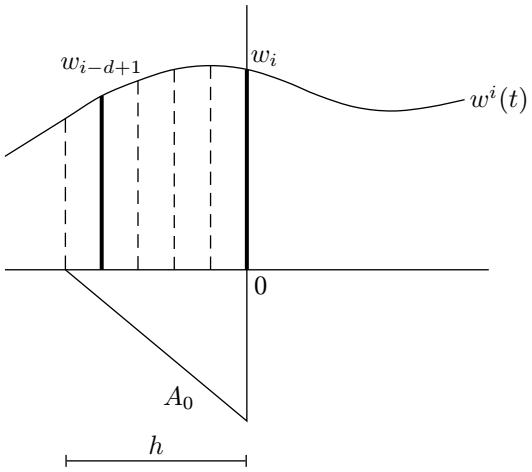
Vzhledem k tomu, že suma $\sum_{k=1}^{h-1} w_{i+k}$ se vyskytuje i ve vztahu (2.48) pro výpočet hodnoty S_L^{i+h}, využijeme její hodnotu dvakrát. Místo předchozího počtu operací $2h-3$ pro výpočet hodnoty S_R^i tak máme pouze 4 aritmetické operace. Tímto jsme dokázali snížení celkového počtu operací nutných pro výpočet hodnoty výstupního signálu $TFF(t_i)$ na 9 elementárních aritmetických operací, čímž se výpočetní složitost stala nezávislou na parametru h.

□

Singulární fuzzy filtr

Všechny doposud uvedené varianty fuzzy filtru prokazovaly jisté časové zpoždění v podobě d hodnot, které musely být načteny dopředu, než bylo možno vyčíslit hodnotu výstupního signálu. V čase $t_i = tT$ tak hodnota výstupního signálu odpovídala hodnotě $w_{i-d} = w((i-d)T)$ originálního signálu. Pro situace, kdy potřebujeme mít okamžitou odezvu na hodnoty w_i, máme možnost určit hodnotu výstupního signálu $FF(t_i)$ pouze na základě již přijatých hodnot $w_0, \ldots, w_i$.

Pro tyto účely může být použit upravený klouzavý filtr, kdy pro výpočet hodnoty komponenty F_0^i jsou použity pouze hodnoty $w_{i-h+1}, \ldots, w_i$. Podle definice 11 se tak komponenta F_0 příslušející k bázické funkci A_0 stává *singulární* komponentou fuzzy rozkladu záporné části časové osy, odkud také pochází název této varianty fuzzy filtru.



Obrázek 2.21: Singulární fuzzy filtr.

Výpočetní složitost singulárního fuzzy filtru odpovídá počtu operací vyhodnocení výrazu:

$$SFF(t_i) = F_0^i = \frac{\sum_{k=1-h}^{0} A_0(t_k) w_{i+k}}{\frac{h}{2}},$$

což odpovídá $2h - 1$ elementárním matematickým operacím.

Lineární fuzzy filtr

Pokud v případě singulárního fuzzy filtru uvažujeme opět pouze trojúhelníkový tvar bázické funkce, můžeme použít obdobný myšlenkový postup při optimalizaci jeho výpočtu, jako tomu bylo v případě trojúhelníkového fuzzy filtru.

Věta 10.
Pokud je v případě singulárního fuzzy filtru použit trojúhelníkový tvar bázické funkce, je nutno k vyhodnocení každé hodnoty výstupního signálu nejvýše 6 elementárních aritmetických operací.

Důkaz 5. Hodnoty bázické funkce můžeme nyní vyjádřit pomocí vztahu:

$$A_0(t_k) = \frac{k}{h} + 1.$$

Pro součet G^i, potřebný k výpočtu hodnoty komponenty $F_0^i = \dfrac{G^i}{h/2}$ tak dostáváme:

$$G^i = \sum_{k=1-h}^{0} A_0(t_k) w_{i+k} = \sum_{k=1-h}^{0} \left(\frac{k}{h} + 1\right) w_{i+k}.$$

V čase t_{i-1} máme:

$$G^{i-1} = \sum_{k=1-h}^{0} A_0(t_k) w_{i-1+k} = \sum_{k=1-h}^{0} \left(\frac{k}{h} + 1\right) w_{i-1+k}.$$

Přeindexováním dostaneme:

$$G^{i-1} = \sum_{k=-h}^{-1} \left(\frac{k+1}{h} + 1\right) w_{i+k},$$

což můžeme převést na tvar:

$$G^{i-1} = \sum_{k=-h}^{-1} \left(\frac{k}{h} + 1\right) w_{i+k} + \frac{1}{h} \sum_{k=-h}^{-1} w_{i+k},$$

ze kterého je již vidět vyjádření následující hodnoty G^i pomocí G^{i-1}:

$$G^i = G^{i-1} + w_i - \frac{1}{h} \sum_{k=-h}^{-1} w_{i+k},$$

což dává 5 elementárních aritmetických operací, když na průběžnou aktualizaci hodnoty sumy $\sum_{k=-h}^{-1} w_{i+k}$ s využitím kruhové fronty počítáme 2 operace. Celkově jsme ukázali, že i se závěrečným dělením hodnotou $\dfrac{h}{2}$ potřebujeme na výpočet hodnoty výstupního signálu $LFF(t_i)$ 6 aritmetických operací.

□

Shrnutí a srovnání

Získané výsledky týkající se výpočetní složitosti jednotlivých variant fuzzy filtrů můžeme pro srovnání shrnout do tabulky 2.7

Všechny výše uvedené varianty fuzzy filtru je také možno přímo použít na filtrování průběhu libovolné jednorozměrné funkce a mělo by být vidět přímočaré zobecnění pro funkce vícerozměrné. Zvláště pro vícerozměrné funkce byla navržena tzv. *líná* varianta inverzního fuzzy filtru.

Líný fuzzy filtr

Líný fuzzy filtr je varianta inverzního fuzzy filtru, která je vhodná v situaci, kdy předem víme, že filtrované hodnoty budeme postupně požadovat pouze v omezeném počtu bodů z definičního

Varianta fuzzy filtru	Časové zpoždění d	Výpočetní složitost
Inverzní fuzzy filtr	$2(h-1)$	7
Klouzavý fuzzy filtr	$h-1$	$2(h+1)$
Trojúhelníkový fuzzy filtr	$h-1$	9
Singulární fuzzy filtr	0	$2h-1$
Lineární fuzzy filtr	0	6

Tabulka 2.7: Porovnání jednotlivých fuzzy filtrů.

oboru originální funkce f. V takovém případě nemusíme počítat hodnoty všech komponent, ale pouze těch, do jejichž oblasti působnosti požadované body padnou.

Na začátku práce s filtrem je v rámci inicializace alokován paměťový prostor na komponenty, přičemž všechny jejich hodnoty jsou nastaveny na speciální hodnotu *undef*. Postupně jak přicházejí požadavky na filtrované hodnoty, se vždy nejprve ověří, zda jsou známy hodnoty všech komponent nutných k výpočtu inverzní F-transformace v daném bodě. V případě, že hodnota některé potřebné komponenty není známa, je spočítána nejprve její hodnota a uložena na odpovídající místo pro případný další požadavek. Během používání filtru tak postupně narůstá počet známých komponent.

2.7 Aplikace F-transformace na řešené úlohy

Některé z výše uvedených algoritmů F-transformace a na ní založených fuzzy filtrů byly implementovány jako součást objektové knihovny `IRAFMlib` napsané v jazyce C++. Následně byly tyto algoritmy použity v těchto aplikacích:

- V části systému LFLC 2000 zabývající se učením a prací s externími daty bylo vytvořeno uživatelské prostředí umožňující výpočet komponent dopředné F-transformace na základě dat ze vstupního souboru, které je možno vzápětí použít pro vyhodnocení hromadné inverzní F-transformace na zadaných vstupních datech.
- Inverzní fuzzy filtr byl použit v systému LFLC 2000 pro úpravu průběhu výsledné funkce inferenčního mechanismu pro libovolnou kombinaci inferenční a defuzifikační metody.
- Varianta klouzavého fuzzy filtru byla použita při implementaci tzv. *hladké dedukce*, která je rovněž součástí softwarového balíku LFLC 2000.
- V rámci účasti v soutěži NN3 byla vytvořena aplikace používající inverzní fuzzy filtr pro analýzu a následnou predikci časových řad. Hlavní myšlenka spočívá v rozložení časové řady na trendovou a aditivní sezónní složku. Právě pro určení trendu časové řady je využito inverzního fuzzy filtru, který má pro tyto účely vhodné vyhlazující vlastnosti. Předpověď trendu časové řady je tak redukována na určení následujících komponent inverzního fuzzy filtru. Jednou z možností pro předpověď komponent je využití logické dedukce vycházející z několika předchozích hodnot. Vektor sezónní složky, v rámci předpokládané periody, je pak jednoduše určen jako lineární kombinace hodnot z několika předchozích period. Pro podrobnější popis celé metody viz [11, 12, 13].

– Dvojrozměrná F-transformace je použita jako základ vyvinuté aplikace pro kompresi obrázků [14, 45, 46]. Jedná se o ztrátovou kompresi, která je založena na myšlence interpretace obrázku jako dvojrozměrné funkce. Do výsledného komprimovaného souboru jsou pak ukládány pouze vhodně zakódované hodnoty komponent získaných aplikací dvojrozměrné F-transformace na tuto funkci.

Kapitola 3

Fuzzy modelování

Pod pojmem *fuzzy modelování* rozumíme modelování vlastností a chování systémů, jejichž některé proměnné nabývají hodnot (stavů) definovaných pomocí slov přirozeného jazyka, jejichž význam modelujeme pomocí fuzzy množin. Tyto proměnné nazýváme *jazykové proměnné*, přičemž operace s nimi prováděné mají základ ve fuzzy logice. Modelem v této souvislosti rozumíme uměle vytvořený systém, který je jistým zjednodušením modelovaného systému (reality), vystihující obraz pouze těch vlastností originálního systému, které jsou pro nás v dané chvíli relevantní z hlediska jejich zkoumání.

3.1 Nástroje pro fuzzy modelování

Základní myšlenkou modelování obecně je, aby nám model systému umožňoval provádění různých experimentů, ať už myšlenkových nebo simulačních, jejichž výsledky mají vypovídající hodnotu o původním modelovaném systému. K tomuto účelu používáme mimo jiné nástroje fuzzy aproximace popsané v předchozí kapitole. Odlišným způsobem práce s dříve uvedenými fuzzy IF–THEN pravidly je přístup tzv. *logické dedukce na základě percepce pozorování* (PbLD – Perception-based Logical Deduction) [21, 22], kdy se na pravidla díváme jako na speciální podmíněné výrazy přirozeného jazyka.

V tomto případě nejsou fuzzy IF–THEN pravidla používána k popisu funkční závislosti, ale jako nástroje pro vyvozování logických důsledků například při rozhodování, klasifikaci objektů, či v oblasti fuzzy regulace k vyjádření popisu regulační strategie. Názvy fuzzy množin vyskytujících se v těchto pravidlech jsou interpretací významu slov přirozeného jazyka – tzv. evaluační jazykové výrazy – jsou to hodnoty, jichž může nabývat výše zmíněná jazyková proměnná, popisující některý dílčí stav systému. Pro naše účely bude dostačující zjednodušená podoba definice jazykové proměnné.

Definice 25 (Jazyková proměnná).
Jazyková proměnná X je dána strukturou $\langle \mathcal{X}, \mathcal{U}, W, \mathscr{L}, \mathcal{M} \rangle$, přičemž význam jednotlivých položek je následující:

- $\mathcal{X}$ představuje jméno jazykové proměnné vystihující její význam (např. výška).
- $\mathcal{U}$ je typové univerzum hodnot, jichž nabývá odpovídající fyzikální proměnná, jejíž jazykovou

reprezentaci modelujeme (např. rozměr v *cm*).

- W je množina *kontextů*. Jazykovým kontextem v této souvislosti rozumíme charakteristiku prostředí, ve kterém se pohybujeme a popis okolností, za kterých hodnoty dané jazykové proměnné můžeme použít.

- $\mathscr{L}$ je formální jazyk, jehož slova představují hodnoty, kterých jazyková proměnná může nabývat – tzv. *jazykové výrazy*. V jednodušších případech může být zadán výčtem všech slov (v případě výšky dospělého člověka např. $\mathscr{L} = \{$*trpaslík, malý, středně vysoký, vysoký, velmi vysoký, obr*$\}$), většinou však k tomu používáme generativní gramatiku G – tzv. *syntaktické pravidlo*, takže jazyk je pak dán jako $\mathscr{L} = L(G)$.

- $\mathcal{M}$ je tzv. *sémantické pravidlo*, které v rámci aktuálního kontextu přiřazuje každému jazykovému výrazu jeho význam v podobě odpovídající fuzzy množiny:

$$\mathcal{M} : \mathscr{L} \times W \longrightarrow \mathcal{F}(\mathcal{U}).$$

Jazykový kontext

Specifikace kontextu je obecně velmi komplexní záležitost, na kterou lze pohlížet z více hledisek a v několika úrovních podle charakteru dané oblasti a také podle toho do jakých podrobností při modelování chceme jít.

Pro naše účely je dostačující mít kontext zadán rozsahem hodnot z univerza $\mathcal{U}$ v podobě *centrovaného intervalu* $w = \langle v_L, v_C, v_R \rangle$, kde interval $\langle v_L, v_R \rangle \subseteq \mathcal{U}$ představuje rozmezí přípustných hodnot a $v_C \in \langle v_L, v_R \rangle$ je typická hodnota, odpovídající jazykovému výrazu „*střední*“ v daném rozmezí přípustných hodnot. Hodnota v_C bývá většinou blíže levé krajní hodnotě, což je dáno částečně i tím, že většina lidských smyslů má logaritmickou statickou charakteristiku danou Weber-Fechnerovým psychofyzikálním zákonem[1], který říká, že mění-li se fyzikální podněty působící na naše smysly řadou geometrickou, vnímáme jejich změnu v řadě aritmetické.

Především pro oblast fuzzy regulace a řízení, kde se nejčastěji pracuje s hodnotami odchylek, které mohou nabývat jak kladných, tak i záporných hodnot, je vhodné zavést pojem *symetrický kontext*. Ten je obecně zadán dvěma centrovanými intervaly $\langle v_L^-, v_C^-, v_R^- \rangle$ a $\langle v_L^+, v_C^+, v_R^+ \rangle$, které splňují podmínku zajišťující spojitost množiny přípustných hodnot dané jazykové proměnné:

$$v_R^- = v_L^+.$$

Fyzikální veličina popisovaná jazykovou proměnnou s takovýmto kontextem pak může nabývat hodnot z intervalu $\left[v_L^-; v_R^+\right]$. Další podmínky obecně nejsou kladeny, avšak v praxi se nejčastěji vyskytuje případ, kdy:

$$v_R^- = v_L^+ = 0$$
$$v_L^- = -v_R^+$$
$$v_C^- = -v_C^+.$$

Syntaktické pravidlo

Typickým syntaktickým pravidlem pro jazykové evaluační výrazy je následující generativní gramatika, jejíž varianta je použita i v softwarovém systému LFLC 2000. Gramatika je popsána

[1] viz `http://cs.wikipedia.org/wiki/Psychologie` odstavec „Obecná charakteristika smyslových orgánů“, nebo: `http://fyzika.jreichl.com/index.php?sekce=browse&page=210`

pomocí BNF s počátečním symbolem ⟨jazykový výraz⟩, přičemž v ní pro jednoduchost není definován neterminál ⟨číselná hodnota⟩, který po přepsání na terminální symboly představuje číselnou hodnotu z intervalu $[v_L; v_R]$.

⟨jazykový výraz⟩::= ⟨jednoduchý výraz⟩
| ⟨fuzzy číslo⟩
| ⟨negovaný výraz⟩
| ⟨složený výraz⟩

⟨negovaný výraz⟩::= *ne* ⟨jednoduchý výraz⟩

⟨složený výraz⟩::= ⟨jazykový výraz⟩ *nebo* ⟨jazykový výraz⟩
| ⟨jazykový výraz⟩ *a* ⟨jazykový výraz⟩

⟨modifikátor⟩::= ⟨zužující modifikátor⟩
| ⟨rozšiřující modifikátor⟩
| ⟨specifikující modifikátor⟩

⟨zužující modifikátor⟩::= *výrazně* | *značně* | *velmi*

⟨rozšiřující modifikátor⟩::= *více méně* | *zhruba* | *dosti zhruba* | *velmi zhruba*
| ⟨prázdný modifikátor⟩

⟨prázdný modifikátor⟩::= ε

⟨specifikující modifikátor⟩::= *spíše*

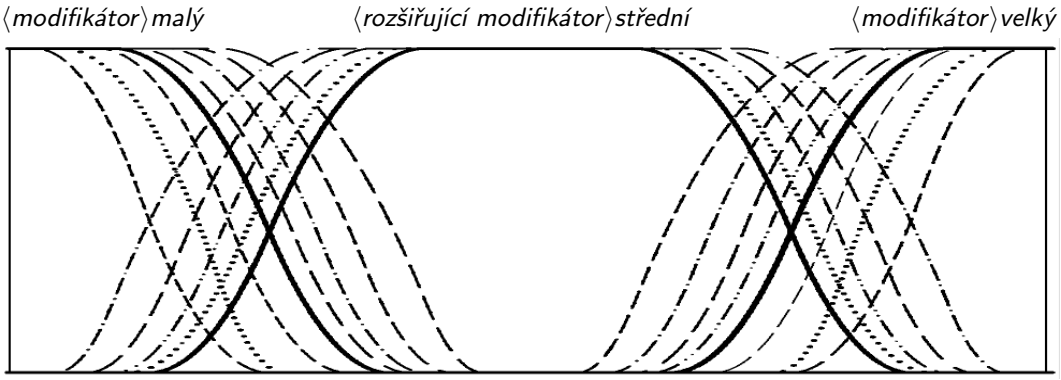
⟨jednoduchý výraz⟩::= ⟨modifikátor⟩ *malý*
| ⟨modifikátor⟩ *velký*
| ⟨rozšiřující modifikátor⟩ *střední*

⟨fuzzy číslo⟩::= ⟨číselný modifikátor⟩⟨číselná hodnota⟩

⟨číselný modifikátor⟩::= *okolo* | ⟨rozšiřující modifikátor⟩

Na obrázku 3.1 jsou pak zobrazeny odpovídající tvary fuzzy množin přiřazené jednoduchým výrazům pomocí sémantického pravidla. Více o struktuře jazykových výrazů a modelování jejich sémantiky s použitím pojmů *intenze* a *extenze* je možno nalézt v [17, 20].

Způsob odvození závěru na základě jazykového popisu a percepce pozorování, která představuje jazykové vyjádření aktuálního vstupu, spočívá nejprve v nalezení nejvhodnějšího pravidla popisujícího aktuální stav, ze kterého je posléze na principu přibližné dedukce odvozen odpovídající závěr. Konkrétní hodnota percepce odpovídající ostrému vstupu u je zjednodušeně řečeno určena jako jazykový evaluační výraz, *vyskytující* se v antecedentových částech pravidel, jemuž je sémantickým pravidlem přiřazena fuzzy množina s nejvyšším stupněm příslušnosti v bodě u. V případě, že takových výrazů existuje více, je z nich jako výsledný vybrán ten nejspecifičtější, což ve většině případů znamená, že má nejmenší základnu.

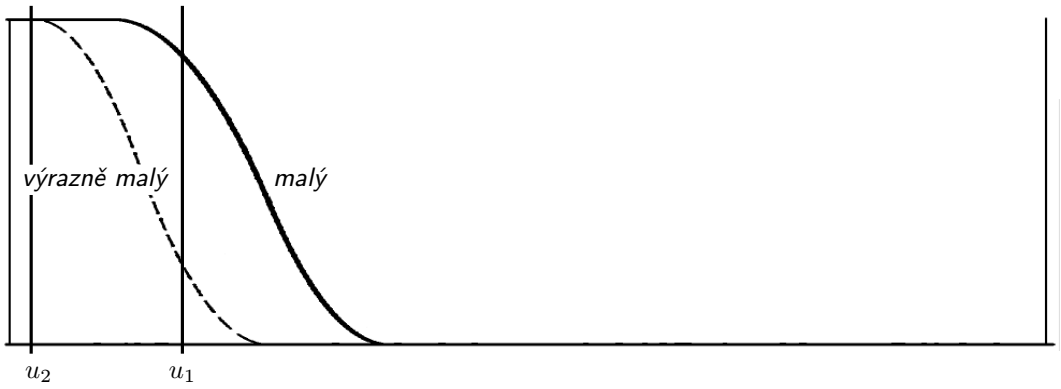


Obrázek 3.1: Typické tvary fuzzy množin přiřazených evaluačním jazykovým výrazům.

Uvažujme například následující jazykový popis:

$$\mathcal{R}_1: \quad \text{IF } X \text{ is } \textit{malý} \text{ THEN } Y \text{ is } \mathscr{B}_1$$
$$\mathcal{R}_2: \quad \text{IF } X \text{ is } \textit{výrazně malý} \text{ THEN } Y \text{ is } \mathscr{B}_2$$

Na obrázku 3.2 jsou zobrazeny dva příklady ostrých vstupů u_1, u_2. Pro vstup u_1 je percepce rovna hodnotě evaluačního jazykového výrazu „malý“, protože stupeň příslušnosti prvku u_1 v odpovídající fuzzy množině je vyšší než u fuzzy množiny přiřazené k výrazu „výrazně malý“. Odvození výstupu bude tedy v tomto případě provedeno podle principu přibližné dedukce na základě pravidla $\mathcal{R}_1$. Pro vstup u_2 jsou stupně příslušnosti v obou fuzzy množinách stejné (rovny jedničce), percepce bude tedy v tomto případě rovna hodnotě výrazu „výrazně malý“, protože blíže specifikuje danou oblast. Pokud by pravidlo $\mathcal{R}_2$ v jazykovém popisu nebylo uvedeno, percepce by byla rovna hodnotě „malý“, což je v souladu s tím, že výrazně malé hodnoty jsou také malé.



Obrázek 3.2: Příklad pro určení percepce pozorování.

Výše několikrát zmiňovaný softwarový systém LFLC 2000 vyvinutý pro podporu fuzzy modelování v sobě obsahuje dříve popsané nástroje pro fuzzy aproximaci i přibližné logické usuzování. Umožňuje efektivně editovat a testovat vytvořené jazykové popisy, přičemž nabízí i několik možností na jejich automatizované vytvoření formou učení z dat [5]. Pro podrobnější popis jeho uživatelského rozhraní a funkcí, které nabízí, viz [1, 2, 3].

Pro modelování složitějších systémů se jeví jako velmi výhodné propojení několika dílčích jazykových popisů do formy hierarchického modelu, jehož prvky mohou být navíc realizovány

různými algoritmy – vytvořenými třeba i na základě *klasických* (ne fuzzy) principů. Takováto dekompozice na dílčí úlohy je vhodná především z pohledu snadnějšího návrhu jazykových pravidel, kdy při více než třech vstupních proměnných se jazykový popis stává kvůli narůstajícímu počtu pravidel méně přehledný a tím pádem i hůře editovatelný.

Jedním z možných využití je například model, kdy fuzzy regulátor hraje roli hlavního řídícího systému v podobě *dohlížecího expertního systému*, který podle konkrétní situace zapojuje v činnost jednotlivé dílčí prvky systému, případně upravuje a nastavuje některé jejich parametry na odpovídající hodnoty podle aktuálních podmínek. Tímto způsobem je možno spojit několik jednoduchých modelů, které popisují systém v okolí určitého pracovního bodu v jeden globální model vytvořený jejich kombinací, podobně jako v případě dříve uvedených Takagi-Sugeno pravidel.

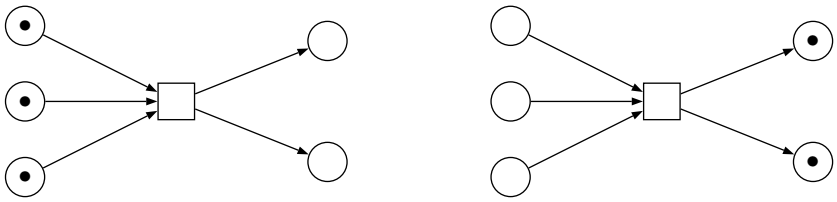
Pro modelování takovýchto hierarchických systémů mohou sloužit Fuzzy Petriho sítě, vzniklé modifikací klasických Petriho sítí, které samy o sobě jsou dostatečně silným nástrojem pro modelování diskrétních systémů.

3.2 Fuzzy Petriho sítě

V této části se zaměříme na využití Petriho sítí pro fuzzy modelování, a to především jako vizualizačního nástroje pro fuzzy IF–THEN pravidla složitějších systémů popsaných pomocí na sebe navzájem navazujících jazykových bází pravidel.

Motivací pro vznik Petriho sítí bylo modelovat řídící systémy. Neformálně je Petriho síť graf se dvěma typy uzlů (místy a přechody) s násobnými hranami. Hrany jsou přitom pouze mezi uzly různých typů. Obecně žádná další omezení nejsou kladena, zejména tedy nemusí být vyváženy vstupní a výstupní stupně uzlů.

Celkový stav systému popisujeme souhrnem jeho dílčích (*parciálních*) stavů, ve kterých se může nacházet. Parciální stavy systému jsou pomocí Petriho sítí modelovány *místy* a možné události nastávající v systému jsou definovány *přechody*. Místa v grafické reprezentaci Petriho sítě značíme pomocí symbolu ○ a přechody pomocí □ .



Obrázek 3.3: Provedení přechodu (výskyt události) v Petriho síti.

Okamžitý stav systému je definován umístěním *značek* (tokens) v místech, což v grafu Petriho sítě vyjadřujeme tečkami v místech ⊙. Přítomnost značky v místě modeluje skutečnost, že daný dílčí stav je momentálně aktuální, resp. podmínka je splněna. Každý přechod má definována vstupní a výstupní místa, což je v grafu Petriho sítě vyjádřeno orientovanými hranami mezi místy a přechody: ○⟶□ a □⟶○ . Tím je dáno, které aspekty stavu systému podmiňují výskyt odpovídající události (provedení přechodu), a které aspekty stavu jsou výskytem této události

ovlivněny. Každý přechod má tedy definovány vstupní a výstupní podmínky. Přechod může být proveden v případě, že všechna jeho vstupní místa obsahují značky, tj. má splněny všechny vstupní podmínky. Provedením přechodu (viz obr. 3.3) se odstraní značky ze vstupních míst (vstupní podmínky přestanou platit) a umístí se nové značky do výstupních míst (uplatní se výstupní podmínky). Provedení přechodu je atomická operace, která odpovídá výskytu události.

Definice 26 (Petriho síť).
Petriho síť je struktura $\mathcal{N} = \langle P, T, F \rangle$, kde

- P je konečná množina *míst* (*places*)
- T je konečná množina *přechodů* (*transitions*), $P \cap T = \emptyset$
- F je *přechodová funkce* (*float*)

$$F : (P \times T) \cup (T \times P) \to \mathbb{N}_0$$

Definice 27.
Nechť $\mathcal{N} = \langle P, T, F \rangle$ je Petriho síť. Pro každý přechod $t \in T$ a pro každé místo $p \in P$ definujeme množiny

$$t^\bullet = \{p \in P | F(t,p) > 0\}$$
$$^\bullet t = \{p \in P | F(p,t) > 0\}$$
$$p^\bullet = \{t \in T | F(p,t) > 0\}$$
$$^\bullet p = \{t \in T | F(t,p) > 0\}$$

Notace se přirozeně rozšiřuje na *množiny* přechodů a míst.

Neformálně můžeme říci, že pro místo $p \in P$ označuje $^\bullet p$ množinu přechodů, z nichž místo p může získat token a $p^\bullet$ je množina přechodů, do nichž může token odevzdat.

Pro pozdější účely můžeme definovat pojmy *vstupní*, *výstupní*, *počáteční* a *koncové* místo.

Definice 28.
Nechť $\mathcal{N} = \langle P, T, F \rangle$ je Petriho síť. Místo $p \in P$ označíme jako

- *vstupní*, pokud platí: $p^\bullet \neq \emptyset$
- vstupní místo přechodu t, pokud $F(p,t) > 0$, resp. $p \in {}^\bullet t$
- *výstupní*, pokud platí: $^\bullet p \neq \emptyset$
- výstupní místo přechodu t, pokud $F(t,p) > 0$, resp. $p \in t^\bullet$
- *počáteční*, pokud $^\bullet p = \emptyset$
- *koncové*, pokud $p^\bullet = \emptyset$

Z definice je zřejmé, že některá místa mohou být zároveň vstupní i výstupní.

Stav systému popsaného Petriho sítí je dán tzv. *značením*.

Definice 29 (Značení).
Nechť $\mathcal{N} = \langle P, T, F \rangle$ je Petriho síť. *Značení* (marking) sítě $\mathcal{N}$ je funkce

$$M : P \to \mathbb{N}_0.$$

Neformálně lze značení chápat jako distribuci tokenů v síti – udává počet tokenů v jednotlivých místech.

Dynamika Petriho sítě spočívá v provádění přechodů. Vliv značení sítě na proveditelnost přechodů a vliv provedení přechodu na značení sítě jsou určeny tzv. *evolučními pravidly*, danými následující definicí. Pro ilustraci slouží obr. 3.4.

Definice 30 (Dynamika Petriho sítě).
Uvažujme síť $\mathcal{N}$ a značení M.

1. Přechod $t \in T$ je *proveditelný* v M, pokud

$$(\forall p \in {}^\bullet t) : M(p) \geq F(p, t).$$

2. Je-li přechod $t \in T$ proveditelný, může být *proveden*, čímž se značení M změní na M' definované takto:

$$(\forall p \in P) : M'(p) = M(p) - F(p, t) + F(t, p).$$

3. Může-li být přechod $t \in T$ ve značení M proveden, přičemž výsledným značením je M', říkáme, že M' je *přímo dostupné* z M *provedením* t, což zapisujeme

$$M[t\rangle M'.$$

Obrázek 3.4: Dynamika Petriho sítě – provedení přechodu t.

Pokud přechody v daném značení nejsou konfliktní (viz dále), mohou být realizovány nezávisle na sobě, v libovolném pořadí (paralelně). Dva současně proveditelné přechody (v daném značení sítě) prohlásíme za konfliktní, když provedení jednoho z nich způsobí, že druhý přestane být proveditelný.

Obecně jde o nedeterministický stroj, neboť v daném značení může být proveditelných více přechodů současně a stroj se tedy může dostat potenciálně do několika dalších stavů. Množina všech možných posloupností změn stavu reprezentuje chování systému. Množina všech možných stavů, do kterých se stroj může dostat z počátečního stavu, je jeho stavový prostor.

3.2.1 Barvená CE-Petriho síť

S nárůstem využití Petriho sítí vznikala potřeba stávající síť upravovat a rozšiřovat tak, aby bylo možno modelovat různé typy problémů v různých oblastech, což vedlo k jejich modifikacím. Pro nás zajímavou modifikací Petriho sítí jsou tzv. *barvené* Petriho sítě (CPN – Coloured Petri Net) ve variantě CE-sítě (Condition-Event Net). Neformálně spočívá rozšíření v tom, že token je nositelem *barvy* (*typu*), čímž je umožněna vyšší flexibilita podmínek kladených na proveditelnost přechodu, přičemž ve variantě CE-sítě připouštíme nejvýše jednu značku v místě.

Definice 31 (Barvená CE-Petriho síť).
Barvená CE-Petriho síť je struktura $\mathcal{N}_C = \langle \Sigma, P, T, F, C \rangle$, kde:

- Σ je konečná množina barev (*typů*)
- P je konečná množina míst
- T je konečná množina přechodů
- F je konečná množina orientovaných hran

$$F \subseteq (P \times T) \cup (T \times P)$$

- C představuje funkci ohodnocení hran

$$C : F \to \Sigma$$

Stav systému popsaného barvenou Petriho sítí je dán, analogicky ke „klasické“ Petriho síti, *barevným značením*.

Definice 32 (Barevné značení).
Nechť $\mathcal{N}_C = \langle \Sigma, P, T, F, C \rangle$ je barvená Petriho síť. *Barevným značením* rozumíme funkci

$$M : P \to \Sigma_0,$$

kde Σ_0 představuje množinu Σ rozšířenou o *prázdný* prvek ε.

Definice 33 (Dynamika barvené CE-Petriho sítě).
Uvažujme barvenou síť $\mathcal{N}$ a její odpovídající značení M.

1. Přechod $t \in T$ je *proveditelný* v M, pokud

   $$(\forall p \in {}^\bullet t) : M(p) = C(p, t)$$

   a zároveň

   $$(\forall p \in t^\bullet) : M(p) = \varepsilon.$$

2. Je-li přechod $t \in T$ proveditelný, může být *proveden*, čímž se značení M změní na M', definované takto:

   $$(\forall p \in {}^\bullet t) : M'(p) = \varepsilon$$

   a

   $$(\forall p \in t^\bullet) : M'(p) = C(t, p).$$

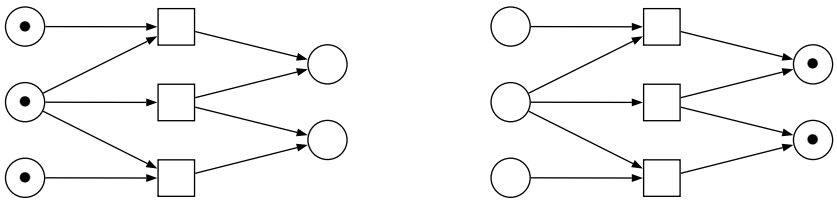
3. Může-li být přechod $t \in T$ ve značení M proveden, přičemž výsledným značením je M', říkáme, že M' je *přímo dostupné* z M *provedením* t, což zapisujeme

$$M[t\rangle M'.$$

Právě uvedenou variantu barvené CE-Petriho sítě je možno přímo použít k modelování a vizualizaci IF–THEN pravidel. Množina Σ v tomto případě představuje konečnou množinu jevů, které v systému mohou nastat. Neformálně řečeno se při použití sítě pro tyto účely v místech vyskytují tokeny nesoucí informaci o nastalých jevech, přičemž hrany jsou ohodnoceny jevy, které musejí nastat pro splnění podmínky proveditelnosti přechodů.

3.2.2 Fuzzy Petriho síť

Pro naše účely modifikujeme barvenou CE-Petriho síť do fuzzy varianty, která se liší především svou dynamikou a přidanou sémantikou. Při použití fuzzy přístupu nastává často situace, kdy je jednou hodnotou (jevem) splněno *více* podmínek v různých stupních. Tomu bude odpovídat odlišná dynamika sítě, při které dojde k provedení *všech* proveditelných přechodů současně (viz příklad na obr 3.5). Pro takový druh dynamiky musí být vyřešeno chování sítě vzhledem k hodnotě výstupního místa aktualizovaného *současně* několika přechody, což úzce souvisí se sémantikou a interpretací samotných IF–THEN pravidel.



(M) (M')

Obrázek 3.5: Dynamika fuzzy Petriho sítě.

Chování sítě tak můžeme sledovat ve dvou rovinách. Jednak je to vlastní dynamika sítě závislá na samotném rozmístění tokenů a za druhé nás zajímá sémantická úroveň pracující s hodnotami tokenů. Při použití inferenční metody typu FITA (First Infer, Then Aggregate) můžeme velmi zhruba a neformálně říci, že tokeny nesou hodnoty představující fuzzy množiny a přechody jsou interpretovány jako fuzzy relace odpovídající fuzzy IF–THEN pravidlu sestavenému z jazykových výrazů, jimiž jsou ohodnoceny vstupní a výstupní hrany přechodu. Agregace výsledků inferencí z jednotlivých pravidel se provádí právě při vyhodnocování hodnoty tokenu výstupního místa z několika jeho vstupních přechodů.

Poznámka 7 (Inference typu FATI).
Při použití inferenční metody typu FATI (First Aggregate Then Infer) je nutno před samotnou aplikací evolučních pravidel fuzzy Petriho sítě sestrojit agregované fuzzy relace přiřazené jednotlivým přechodům celého tzv. *clusteru* přechodů a míst.

Definice 34 (Fuzzy Petriho síť).
Fuzzy Petriho síť (FPN) je struktura $\mathcal{N}_F = \langle P, T, F, C, L\rangle$, kde L je zobrazení přiřazující každému

místu $p \in P$ jazykovou proměnnou $X_p = \langle \mathcal{X}_p, \mathcal{U}_p, w_p, \mathscr{L}_p, \mathcal{M}_p \rangle$, a když pomocí Σ označíme sjednocení všech jazykových hodnot příslušejících všem místům $\Sigma = \bigcup_{p \in P} \mathscr{L}_p$, tak struktura $\langle \Sigma, P, T, F, C \rangle$ tvoří barvenou CE-Petriho síť, pro kterou navíc platí:

$$(\forall t \in T)(\exists p, q \in P) : \{(p,t),(t,q)\} \subseteq F,$$

tedy, že každý přechod má alespoň jednu vstupní a jednu výstupní hranu. Dále jsou kladeny doplňující podmínky na funkci ohodnocení hran v tom smyslu, že hrany jsou ohodnoceny pomocí prvků z množiny jazykových výrazů odpovídajících místu, se kterým je hrana spojená.

$$(\forall (p,t) \in F, p \in P, t \in T) : C(p,t) \in \mathscr{L}_p$$

$$(\forall (t,p) \in F, p \in P, t \in T) : C(p,t) \in \mathscr{L}_p.$$

Definice 35 (Fuzzy značení).
Nechť $\mathcal{N}_F = \langle P, T, F, C, L \rangle$ je fuzzy Petriho síť. *Fuzzy značením* rozumíme funkci M přiřazující každému místu p fuzzy množinu nad univezem $\mathcal{U}_p$ příslušejícím dané jazykové proměnné s tímto místem svázané podle funkce L.

Poznámka 8.
Pomocí *prázdné množiny* přiřazené místu ve fuzzy značení dáváme najevo nepřítomnost tokenu. Všechny ostatní hodnoty znamenají přítomnost tokenu nesoucího hodnotu přiřazené fuzzy množiny.

Definice 36 (Dynamika fuzzy Petriho sítě).
Nechť M je fuzzy značení sítě $\mathcal{N}_F$.

1. Přechod $t \in T$ je *proveditelný* v M, pokud

$$(\forall p \in {}^\bullet t) : M(p) \neq \emptyset.$$

2. Na sémantické úrovni každému přechodu t přísluší IF–THEN pravidlo $\mathcal{R}_t$ ve tvaru:

$$\mathcal{R}_t : \mathsf{IF}\ X_1\ \mathsf{is}\ C(p_1,t)\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ C(p_m,t)$$
$$\mathsf{THEN}\ Y_1\ \mathsf{is}\ C(t,q_1)\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ Y_n\ \mathsf{is}\ C(t,p_n).$$

   Pomocí $\mathcal{R}_t^{q_j}$ budeme značit pravidlo ve tvaru:

$$\mathcal{R}_t : \mathsf{IF}\ X_1\ \mathsf{is}\ C(p_1,t)\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ C(p_m,t)\ \mathsf{THEN}\ Y_j\ \mathsf{is}\ C(t,q_j),$$

   kde

   - $p_1, \ldots, p_m$ jsou vstupní místa přechodu t: $\{p_1, \ldots, p_m\} = {}^\bullet t$
   - $q_1, \ldots, q_n$ jsou výstupní místa přechodu t: $\{q_1, \ldots, q_n\} = t^\bullet$
   - $X_1, \ldots, X_m$ jsou jazykové proměnné svázané s místy $p_1, \ldots, p_m$
   - $Y_1, \ldots, Y_n$ jsou jazykové proměnné svázané s místy $q_1, \ldots, q_n$
   - $C(p_1,t), \ldots, C(p_m,t), C(t,q_1), \ldots, C(t,q_n)$ jsou hodnoty jazykových výrazů z množin $\mathscr{L}_{p_1}, \ldots, \mathscr{L}_{p_m}, \mathscr{L}_{q_1}, \ldots, \mathscr{L}_{q_n}$ přiřazené hranám $(p_1,t), \ldots, (p_m,t), (t,q_1), \ldots, (t,q_n)$.

3. Nechť T_M je množina všech proveditelných přechodů. Je-li množina T_M *neprázdná*, pak pomocí $M\,[\rangle\, M'$ zapisujeme *přechod* od značení M ke značení M', které je definované takto:

$$(\forall p \in ({}^\bullet T_M \setminus T_M^\bullet)) : M'(p) = \emptyset$$

a

$$(\forall q \in T_M^\bullet) : M'(q) = \mathrm{INF}(\{\mathcal{R}_{t_i}^q\}_{i=1}^n, [M(p_1), \ldots, M(p_m)]),$$

kde

- $\{\mathcal{R}_{t_i}^q\}_{i=1}^n$ je systém IF–THEN pravidel příslušejících přechodům $t_1, \ldots, t_n$ pro výstupní místo q, přičemž $\{t_1, \ldots, t_n\} = {}^\bullet q$.
- $[M(p_1), \ldots, M(p_m)]$ je vektor fuzzy množin přiřazených značením M místům $p_1, \ldots, p_m$, kde $\{p_1, \ldots, p_m\} = {}^{\bullet\bullet} q$.
- $\mathrm{INF}(\{\mathcal{R}_{t_i}^q\}_{i=1}^n, [M(p_1), \ldots, M(p_m)])$ je výsledná fuzzy množina získaná inferencí z báze pravidel $\{\mathcal{R}_{t_i}^q\}_{i=1}^n$ pro vstupy $X_1 = M(p_1), \ldots, X_m = M(p_m)$.

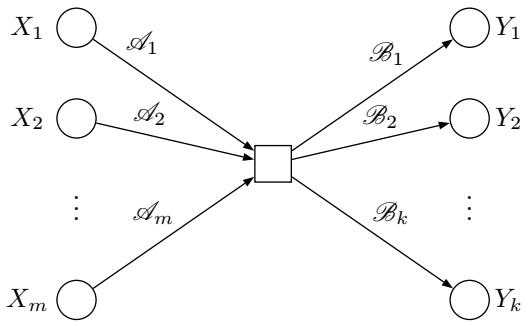
Poznámka 9 (Inferenční metoda).
Pro jednoduchost předpokládáme, že v celé síti je pro sémantickou úroveň týkající se výpočtu hodnot tokenů ve výstupních místech používána stejná inferenční metoda.

3.2.3 Modelování IF–THEN pravidel pomocí fuzzy Petriho sítí

Nyní můžeme přímočaře použít fuzzy Petriho sítě k modelování a vizualizaci IF–THEN pravidel. Každá jazyková proměnná je svázána s jedním místem fuzzy Petriho sítě. Pravidlo tvaru

$$\mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ \mathscr{A}_m\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathscr{B}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ Y_k\ \mathsf{is}\ \mathscr{B}_k$$

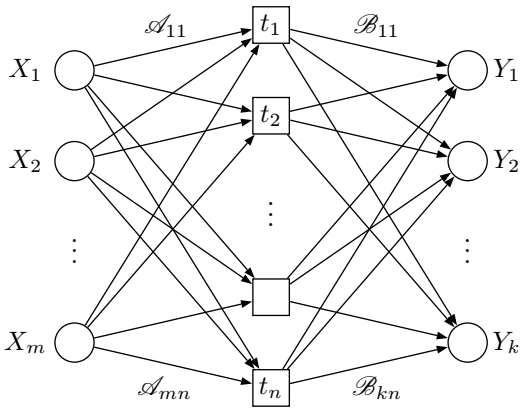
můžeme vyjádřit následující fuzzy Petriho sítí:



Pro soustavu pravidel ve tvaru

$$\begin{aligned}
\mathcal{R}_1 :&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_{11}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ \mathscr{A}_{m1}\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathscr{B}_{11}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ Y_k\ \mathsf{is}\ \mathscr{B}_{k1}\\
\mathcal{R}_2 :&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_{12}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ \mathscr{A}_{m2}\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathscr{B}_{12}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ Y_k\ \mathsf{is}\ \mathscr{B}_{k2}\\
\vdots\quad &\quad\vdots\quad\quad\vdots\quad\quad\vdots\quad\quad\vdots\quad\quad\vdots\quad\quad\vdots\quad\quad\vdots\\
\mathcal{R}_n :&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_{1n}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_m\ \mathsf{is}\ \mathscr{A}_{mn}\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathscr{B}_{1n}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ Y_k\ \mathsf{is}\ \mathscr{B}_{kn}
\end{aligned}$$

sestrojíme síť:



3.2.4 Konstrukce znalostní báze

Fuzzy Petriho sítě, jak byly definovány v předchozí části, představují jednak nástroj pro modelování a vizualizaci fuzzy IF–THEN pravidel, ale také díky své sémantice umožňují provádění dedukčního vyvozování na základě vstupních hodnot daných konkrétním fuzzy značením.

Pro praktické účely, především pro využití v jiných aplikacích, je však vhodné mít nástroj, který k vytvořené fuzzy Petriho síti dokáže zkonstruovat odpovídající znalostní bázi. Tato báze sestává obecně z několika navzájem na sebe různě navazujících jazykových popisů.

Z těchto důvodů byl vytvořen softwarový nástroj pro převod fuzzy Petriho sítě na síť vzájemně propojených jazykových popisů (LFLN – Linguistic Fuzzy Logic Net). Vzhledem k tomu, že jsme měli k dispozici inferenční mechanismus zabudovaný v softwarovém balíku LFLC 2000, přizpůsobili jsme konverzní nástroj jeho potřebám, abychom jej mohli použít jako výkonného jádra provádějícího vlastní inferenci.

Toto přizpůsobení se týká především omezení kladených na stavbu jazykového popisu tak, aby jej bylo možno zpracovat systémem LFLC 2000. V současné podobě systém LFLC 2000 pracuje s jazykovými popisy

1. které mají *jedinou* sukcedentovou proměnnou a

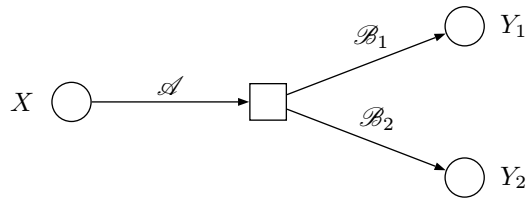
2. u kterých se v antecedentové části pravidel musejí vyskytovat *všechny* vstupní proměnné jazykového popisu.

První podmínku lze snadno splnit rozložením případné soustavy pravidel s více sukcedentovými proměnnými na několik jazykových popisů – pro každou výstupní proměnnou jeden. Co se týče druhé podmínky, nabízí systém LFLC 2000 řešení pomocí speciálního jazykového výrazu UNDEF[2], který způsobí, že na hodnoty této proměnné se v rámci posuzování aplikovatelnosti daného pravidla, ve kterém se výraz vyskytuje, nebude brát zřetel.

Vše můžeme nejlépe demonstrovat na několika jednoduchých příkladech.

Příklad 1.
K fuzzy Petriho síti:

[2]Např. X_1 is UNDEF.

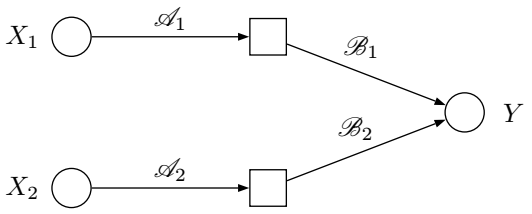


sestrojíme znalostní bázi sestávající ze dvou jazykových popisů:

1.
$$\mathcal{R}_1 : \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathscr{B}_1$$

2.
$$\mathcal{R}_1 : \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}\ \mathsf{THEN}\ Y_2\ \mathsf{is}\ \mathscr{B}_2$$

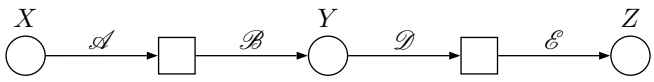
Příklad 2.
K fuzzy Petriho síti:



bude sestrojen jeden jazykový popis obsahující dvě pravidla:

$$\mathcal{R}_1 : \quad \mathsf{IF}\ X_1\ \mathsf{is}\ \mathscr{A}_1\ \mathsf{AND}\ X_2\ \mathsf{is\ UNDEF}\ \ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}_1$$
$$\mathcal{R}_2 : \quad \mathsf{IF}\ X_1\ \mathsf{is\ UNDEF\ AND}\ X_2\ \mathsf{is}\ \mathscr{A}_2\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}_2$$

Příklad 3.
V této situaci je jedno místo zároveň vstupním i výstupním místem dvou přechodů:

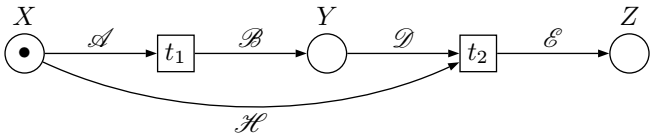


Sestrojíme znalostní bázi sestávající ze dvou jazykových popisů:

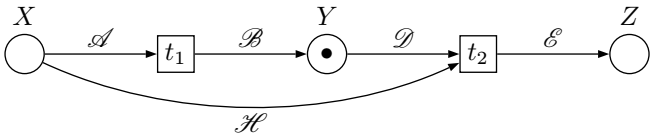
1.
$$\mathcal{R}_1 : \mathsf{IF}\ X\ \mathsf{is}\ \mathscr{A}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathscr{B}$$

2.
$$\mathcal{R}_1 : \mathsf{IF}\ Y\ \mathsf{is}\ \mathscr{D}\ \mathsf{THEN}\ Z\ \mathsf{is}\ \mathscr{E}$$

Poznámka 10.
Na chvíli se ještě u právě uvedeného příkladu pozastavme. Je nutno poznamenat, že se v této situaci můžeme dostat do potíží v případě, že bychom měli hranou propojeny místa s přechody různých „úrovní“. Tedy například v případě sítě:



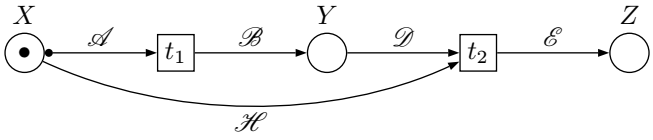
můžeme jazykové popisy zkonstruovat, ale chování sítě jim nebude odpovídat, protože token umístěný v místě X bude „spotřebován“ na provedení přechodu t_1, čímž se síť dostane do stavu:



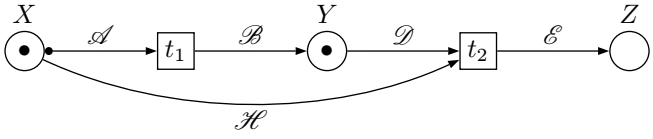
ve kterém již není možno pokračovat žádným přechodem.

Tento problém ponechme prozatím otevřený, s tím, že v dalším budeme předpokládat, že tato situace nenastává. Jedním z možných nabízejících se řešení tohoto problému je rozšíření vyjadřovacích prostředků fuzzy Petriho sítě o tzv. *zachovávající* hranu, která ponechává token v místě i když je proveden přechod s touto s hranou spojený, což značíme pomocí:

Tímto bychom mohli uvedený příklad vyřešit následujícím způsobem:

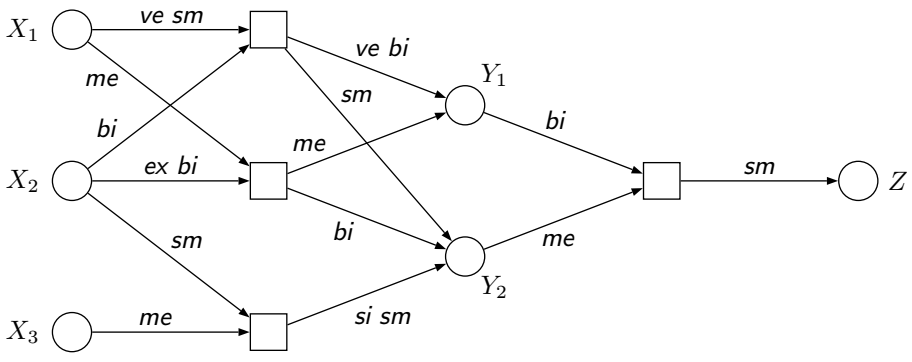


kdy po provedení přechodu t_1 se síť dostane do stavu:



ze kterého může dále pokračovat provedením přechodu t_2.

Příklad 4.
Jako poslední příklad uveďme komplexnější fuzzy Petriho síť ohodnocenou konkrétními jazykovými výrazy:



V tomto případě bude sestavena znalostní báze skládající se z tří jazykových popisů:

1.

$\mathcal{R}_1$: IF X_1 is *ve sm* AND X_2 is *bi* THEN Y_1 is *ve bi*
$\mathcal{R}_2$: IF X_1 is *me* AND X_2 is *ex bi* THEN Y_1 is *me*

2.

$\mathcal{R}_1$: IF X_1 is *ve sm* AND X_2 is *bi* AND X_3 is UNDEF THEN Y_2 is *sm*
$\mathcal{R}_2$: IF X_1 is *me* AND X_2 is *ex bi* AND X_3 is UNDEF THEN Y_2 is *bi*
$\mathcal{R}_3$: IF X_1 is UNDEF AND X_2 is *sm* AND X_3 is *me* THEN Y_2 is *si sm*

3.

$\mathcal{R}_1$: IF Y_1 is *bi* AND Y_2 is *me* THEN Z is *sm*

3.2.5 Dekompozice fuzzy Petriho sítě

Nyní můžeme uvést algoritmus, který na svém vstupu očekává fuzzy Petriho síť, ve které se nevyskytují situace popsané v poznámce 10 a jako výstup vrátí zkonstruovanou znalostní bázi obsahující jazykové popisy sestavené na základě struktury vstupní fuzzy Petriho sítě.

Algoritmus 4: Dekompozice fuzzy Petriho sítě na soustavy jazykových popisů

označení : DFPN

vstup : fuzzy Petriho síť: $\mathcal{N}_F = \langle \mathscr{L}, P, T, F, C \rangle$

výstup : množina jazykových popisů (znalostní báze): KB

$KB := \emptyset$;

// pro každé výstupní místo vytvoříme samostatný jazykový popis

foreach $q \in P$: ${}^\bullet q \neq \emptyset$ **do**

// vytvoříme množinu míst, na kterých je q závislé

$inputs := \emptyset$;

// pro všechny vstupní přechody místa q

foreach $t \in {}^\bullet q$ **do**

// přidáme všechna vstupní místa přechodu t

$inputs := inputs \cup {}^\bullet t$;

end

// vytvoříme jazykový popis (množinu pravidel - RuleBase)

$RB := \emptyset$;

// pro všechny vstupní přechody místa q

foreach $t \in {}^\bullet q$ **do**

// sestavíme odpovídající pravidlo $\mathcal{R}$

$\mathcal{R} :=$ "";

foreach $p \in inputs$ **do**

if $\mathcal{R} \neq$ "" **then** $\mathcal{R} := \mathcal{R} +$ " AND ";

if $p \in {}^\bullet t$ **then**

$\mathcal{R} := \mathcal{R} + p +$ " is " $+ C(p,t)$;

else

$\mathcal{R} = \mathcal{R} + p +$ " is UNDEF ";

end

end

$\mathcal{R} := \mathcal{R} +$ " THEN " $+ q +$ " is " $+ C(t,q)$; // Sukcedentová část pravidla

$RB := RB \cup \mathcal{R}$; // vytvořené pravidlo přidáme do jazykového popisu

end

$KB := KB \cup RB$; // přidáme jazykový popis do znalostní báze

end

return KB;

Uvedený algoritmus DFPN byl implementován v softwarovém nástroji `fpn2lfln`, který jako vstup očekává soubor s Petriho sítí uloženou ve formátu PNML (Petri Net Markup Language) [36, 37]. Jedná se o obecný formát založený na XML [39], určený pro popis Petriho sítě, který je kompatibilní s nástrojem PNK 2.0 (The Petri Net Kernel) [38], ve kterém je možno Petriho síť vizuálně navrhnout. Více zdrojů informací je možno najít na stránkách:

`http://www2.informatik.hu-berlin.de/top/pnk/index.html`

`http://www2.informatik.hu-berlin.de/top/pnml/about.html`

Vytvořený nástroj `fpn2lfln` vnitřně převádí *acyklickou* fuzzy Petriho síť na soustavy jazykových pravidel tvořících znalostní bázi. Poté podle značení počátečních míst, které je rovněž načítáno ze vstupního souboru obsahujícího popis fuzzy Petriho sítě, provádí postupné aktualizace hodnot výstupních jazykových proměnných až po jazykové proměnné asociované s koncovými místy. Na závěr předá program hodnoty všech jazykových proměnných jako výstup. Vnitřní výpočty týkající se hodnot jazykových proměnných jsou prováděny voláním inferenční metody implementované v COM objektu `RuleBaseCOM` ze softwarového balíku LFLC 2000.

Kapitola 4

Fuzzy regulace a řízení

Metody a nástroje fuzzy modelování lze s úspěchem použít pro fuzzy regulaci a řízení technologických procesů. Původní myšlenka využití fuzzy přístupů k řízení systémů pochází od Zadeha [28] a Mamdaniho [29]. Řídicí algoritmus je reprezentován soustavou jazykových pravidel odrážejících jeho řídicí strategii. Hlavní výhodou takového přístupu je možnost implementace heuristických pravidel získaných ze zkušenosti a pomocí intuice jako součást modelu řídicího procesu.

Reálné systémy mohou být obecně velice komplexní a mohou obsahovat několik vzájemně se ovlivňujících veličin, přičemž ne vždy je zcela zřejmá přesná matematická závislost mezi vstupními hodnotami a samotným chováním systému. Z hlediska fuzzy regulace se na *regulovanou soustavu* (proces) díváme jako na „černou skříňku“, u které nás zajímá čistě vstupně/výstupní chování. V dalším budeme předpokládat, že počet vstupních a výstupních proměnných regulované soustavy zůstává v čase konstantní.

Pro zjištění *aktuálního stavu* systému většinou používáme různých fyzikálních měřicích senzorů, které nám dávají informaci o aktuálním tlaku, teplotě, vzdálenosti, atd. v systému. Tyto veličiny představují *výstupní vektor* hodnot $\bar{y} = [y_1, \ldots, y_n]$ z procesu. U některých z těchto hodnot bychom si přáli, aby se v čase vyvíjely podle našich požadavků. Nejčastější je případ, kdy požadujeme, aby se některá veličina y_i ustálila na předem dané, v nějakém časovém intervalu konstantní hodnotě w_i (tzv. *setpoint*), obecně však můžeme požadovat libovolný časový průběh $w_i(t)$. Pomocí hodnoty veličiny w_i tak z vnějšku můžeme ovlivňovat chování regulované soustavy, proto ji také označujeme jako *řídicí veličinu*. Je zřejmé, že obecně může být řídicích veličin více, v takovémto případě hovoříme o tzv. *řídicím vektoru*, který udává *požadovaný stav*.

4.1 Princip fuzzy regulace

Na základě informací o aktuálním a požadovaném stavu regulátor vyhodnotí *akční zásah* v podobě hodnoty *akční veličiny* u, resp. akčního vektoru $\bar{u} = [u_1, \ldots, u_m]$ v případě m akčních veličin. Akční zásah je realizován nastavením vstupních veličin regulované soustavy na hodnoty dané vektorem $\bar{u}$. Konkrétně se může jednat například o otevření či přiškrcení nějakého elektricky řízeného ventilu na požadovanou úroveň apod. Z matematického hlediska tak fuzzy regulátor vlastně realizuje funkci

$$\bar{u}_t = u(\bar{y}_t, \bar{w}_t, t), \tag{4.1}$$

která je vytvořena na základě fuzzy IF–THEN pravidel tvaru.

$$\mathsf{IF}\ (\text{stav procesu})\ \mathsf{THEN}\ (\text{akční zásah do systému}) \tag{4.2}$$

Na proces se ze stejného úhlu pohledu můžeme dívat jako na realizaci funkce $\bar{y}_t = y(\bar{u}_t, t)$, která udává tzv. přenosovou charakteristiku procesu.

Hlavní výhodou fuzzy regulace je, že výše uvedenou funkci y nemusíme znát přesně, čímž odpadá mnohdy složitá *identifikace systému*, která navíc sama o sobě nemusí být zárukou úspěšné regulace. Pro návrh fuzzy regulátoru je postačující znalost *regulační strategie*, mnohdy i v dosti hrubé podobě, např. ve formě znalosti vyjádřené slovy jak *hodně* musíme brzdit, abychom při řízení auta stihli *včas* zastavit.

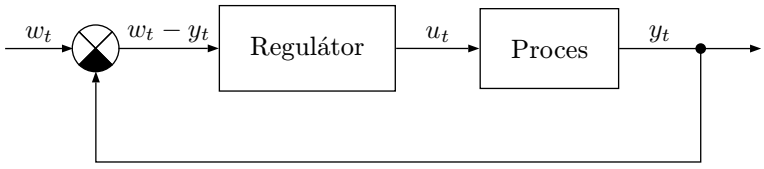
Je zřejmé, že bez újmy na obecnosti se v dalším zkoumání můžeme omezit pouze na regulátory s jedinou výstupní veličinou u, protože v případě procesu s m vstupními veličinami můžeme zkonstruovat m regulátorů fungujících paralelně, nezávisle na sobě.

V rámci snadnějšího a přehlednějšího návrhu budoucího regulátoru je vhodné už při počátečním návrhu reálného systému myslet na jeho regulaci a snažit se, aby jej bylo možno z hlediska řízení dekomponovat na dílčí systémy, které lze regulovat samostatně. Jedná se především o počty vstupních proměnných vstupujících do fuzzy regulátoru, protože, jak ještě bude ukázáno později, jejich počet pro sestavení báze pravidel může vzrůst tím, že často potřebujeme kromě hodnoty samotné i její derivaci, resp. diferenci, případně další odvozené hodnoty.

Dále budeme pro jednoduchost předpokládat, že regulovaná soustava má jednu vstupní veličinu u a jednu výstupní veličinu y, kterou se snažíme regulovat na hodnotu w. Hodnoty těchto veličin v čase t budeme značit pomocí u_t, y_t, w_t.

Základem všech regulačních a řídicích systémů je regulace v uzavřené zpětnovazební smyčce, nejčastěji podle odchylek $e_t = w_t - y_t$ od požadovaného stavu, jak je znázorněno na obr. 4.1. V případě spojitých systémů musíme nejprve určit vzorkovací frekvenci f, s jakou budeme hodnoty v regulačním obvodu aktualizovat. Tím je určena časová perioda $T = 1/f$, která udává interval mezi jednotlivými aktualizačními cykly. Každý průběh tohoto cyklu v čase t je složen z:

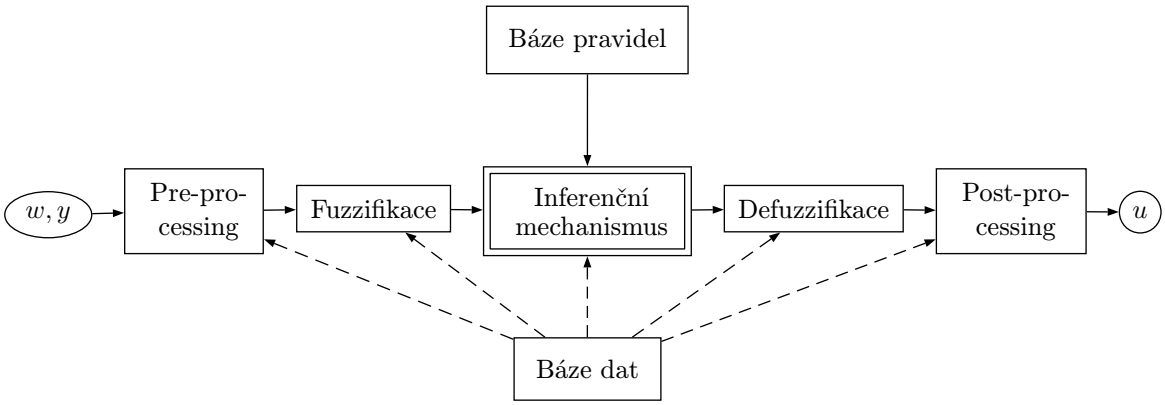
1. Načtení aktuálního stavu regulované soustavy v podobě naměřené hodnoty y_t.
2. Určení odchylky od požadovaného stavu $e_t = w_t - y_t$.
3. Získání hodnoty akčního zásahu regulátoru u_t.
4. Aplikace akčního zásahu nastavením akční veličiny na hodnotu u_t.



Obrázek 4.1: Schéma regulačního obvodu – uzavřená zpětnovazební smyčka.

4.2 Struktura fuzzy regulátoru

Obecné schéma fuzzy regulátoru je uvedeno na obr. 4.2. Vlastní fuzzy regulátor je složen z modulu fuzzifikace, který převádí ostré hodnoty veličin z regulačního obvodu na fuzzy množiny. Tyto fuzzifikované hodnoty vstupují do inferenčního mechanismu, který tvoří jádro fuzzy regulátoru. Ten na základě znalostní báze, tvořené bází dat a především bází pravidel, provádí nejdůležitější část výpočtu akčního zásahu. Výstupem z inferenčního mechanismu je akční zásah v podobě fuzzy množiny, který se v modulu defuzzifikace převádí na ostrou hodnotu, kterou je dále možno použít v regulačním obvodu. Před a za fuzzy regulátor jsou ještě umístěny bloky pre a post-processingu, které zpracovávají hodnoty do potřebné podoby, jak bude popsáno dále.



Obrázek 4.2: Blokové schéma fuzzy regulátoru.

Modul pre-processingu
Jeho úkolem je před-zpracovat data v podobě vstupních signálů získaných z měřicích senzorů a vytvořit z nich číselné hodnoty, případně je ještě nějakým vhodným způsobem upravit. Nejčastější úpravy relizované tímto modulem jsou:

- Kvantování dané přesností měření a ve spojitosti s časováním měření podle dané vzorkovací periody.
- Převod vstupních fyzikálních veličin do číselné reprezentace pomocí tzv. AD převodníku, přičemž často se zároveň provádí i *normalizace*, tj. přepočet do normalizovaného univerza pomocí daného fyzikálního měřítka.
- Vyrovnání předem známých nelinearit daných především charakterem měřené veličiny, případně měřicího senzoru. Může se například jednat o přepočet logaritmické charakteristiky na lineární měřítko apod.
- Filtrování signálu pro odstranění šumu.
- Kombinace několika měření pro získání *klíčových indikátorů*.
- Diferenciace a integrace, resp. jejich aproximace v diskrétních časových intervalech.

Na chvíli se pozastavme u posledního uvedeného bodu. Jak bude uvedeno za chvíli, fuzzy regulátory pracují často kromě samotné odchylky $e_t = w_t - y_t$ v čase t, také s odvozenými hodnotami

odrážejícími dynamiku procesu. Nejčastější odvozené hodnoty, jejichž použití bude vysvětleno dále jsou tyto:

– *Změna odchylky* Δe_t:

$$\Delta e_t = e_t - e_{t-1}, \tag{4.3}$$

kde e_{t-1} představuje odchylku naměřenou v předchozím měřeném časovém okamžiku. Z praktického hlediska, kvůli budoucí možné změně vzorkovací frekvence, je vhodnější pracovat s aproximací její první derivace $\dot{e}$, představující *rychlost změny* odchylky:

$$\dot{e}_t = \frac{\mathrm{d}e}{\mathrm{d}t}(t) \approx \frac{\Delta e_t}{T}, \tag{4.4}$$

kde $T = \dfrac{1}{f}$ je *vzorkovací perioda* představující časový interval mezi jednotlivými měřeními. Pokud totiž používáme v IF–THEN pravidlech jazykovou proměnnou zastupující hodnoty vypočtené podle (4.3) a dojde ke změně vzorkovací frekvence, musí se tomu odpovídajícím způsobem změnit kontext dané jazykové proměnné. Kdežto při použití hodnoty vypočtené podle (4.4) může zůstat kontext odpovídající proměnné zachován a změna hodnoty vzorkovací frekvence by téměř neměla mít zásadní vliv na chování regulace. Toto by samozřejmě byl ideální případ, ale (nejen) vzhledem k tomu, že pracujeme pouze s aproximací, musíme někdy při změně vzorkovací frekvence upravit i samotná pravidla v bázi pravidel.

Další výhodou při použití aproximace rychlosti oproti prosté diferenci jsou také nižší nároky kladené na přesnost vzorkovací periody, což zvláště při vysokých frekvencích může být zásadní.

– *Druhá diference odchylky* $\Delta^2 e_t$:

$$\Delta^2 e_t = \Delta e_t - \Delta e_{t-1},$$

přičemž z obdobných důvodů jako v předešlém případě je výhodnější pracovat s aproximací druhé derivace odchylky v čase t:

$$\ddot{e}_t = \frac{\mathrm{d}^2 e}{\mathrm{d}t^2}(t) \approx \frac{\Delta^2 e_t}{T^2},$$

kterou můžeme interpretovat jako *zrychlení* odchylky, resp. chyby.

– *Kumulativní součet odchylky* δe_t:

$$\delta e_t = \sum_{i=1}^{t} e_i,$$

resp. aproximace hodnoty integrálu

$$\int_0^t e \,\mathrm{d}t \approx \sum_{i=1}^{t} e_i T,$$

která má opět výhodu v tom, že je odolnější vůči změně vzorkovací frekvence.

– Někdy je pro regulaci zapotřebí znát zpětně některou z výše uvedených hodnot pocházející z předchozích měření, obecně v čase $t - k$. V takovém případě tento modul udržuje požadované hodnoty v paměti pro pozdější využití, nejčastěji ve vhodně implementované kruhové vyrovnávací paměti (*buffer*), resp. frontě.

Konečnou úlohou modulu pre-processingu je předat k dalšímu zpracování hodnotu aktuální odchylky $e_t = w_t - y_t$ a podle potřeby další od ní odvozené hodnoty.

Modul fuzzifikace

V modulu fuzzifikace se převádějí předzpracované ostré hodnoty na fuzzy množiny. Pokud jsou měřené hodnoty zatíženy neurčitostí, můžeme tento fakt zohlednit tvarem přiřazené fuzzy množiny. Nejčastější způsob fuzzifikace však bývá proveden vytvořením tzv. *singletonové* fuzzy množiny obsahující pouze jeden prvek se stupněm příslušnosti 1. V podstatě se jedná pouze o trik umožňující provedení přibližné dedukce, jejímž vstupem pro antecedent jsou fuzzy množiny. Z praktického hlediska je kvůli urychlení výpočtu výhodnější singletonový fuzzifikátor vůbec nepoužívat a raději upravit inferenční mechanismus tak, aby jako vstup očekával přímo ostré hodnoty a pracovat s nimi obdobně jak bylo ukázáno při popisu Mamdani-Assilianovy metody na obr. 2.5. Tedy místo hledání obrazu vstupního singletonu v relaci přiřazené jazykovému popisu podle vztahu (2.11), použít pro získání výstupu přímo řez této relace podle vztahu (2.12).

Báze dat

Poskytuje nezbytné informace pro funkci ostatních souvisejících modulů:

- Pro moduly pre a post-processingu poskytuje informaci o rozsahu (měřítku) jednotlivých vstupních a výstupních fyzikálních veličin.
- Pro moduly fuzzifikace a defuzzifikace poskytuje informace o kontextech jazykových proměnných představujících univerza příslušných fuzzy množin.
- Pro inferenční mechanismus obsahuje informaci o tvaru fuzzy množin reprezentujících slovní hodnoty jazykových proměnných, které jsou použity v pravidlech.

Báze pravidel

Společně s bází dat tvoří tzv. *znalostní bázi* fuzzy regulátoru. Obsahuje nejdůležitější informaci pro inferenční mechanismus ve formě fuzzy IF–THEN pravidel, jejichž obecný tvar byl uveden v (4.2), popisujících řídicí strategii. Pravidla jsou většinou zadávána expertem, který ví jak daný systém řídit způsobem, který jej převede do požadovaného stavu.

Inferenční mechanismus

Inferenční mechanismus představuje vlastní výkonné jádro fuzzy regulátoru a je realizován jednou ze dvou dříve uvedených metod interpretace jazykového popisu – buď jako fuzzy aproximace nebo jako logická dedukce na základě percepce pozorování. Na základě báze znalostí a naměřených hodnot se provede přibližná dedukce, jejímž výsledkem je výstupní fuzzy množina $\tilde{u}$, která je dále zpracována v modulu defuzzifikace.

Modul defuzzifikace

Výstupem předchozího inferenčního mechanismu je fuzzy množina $\tilde{u}$, ale akční zásah do řízeného systému musí být ostrá hodnota. Úkolem defuzzifikačního modulu je určení jedné konkrétní ostré hodnoty k fuzzy množině $\tilde{u}$.

Modul post-processingu

Velmi zhruba můžeme říct, že úloha modulu post-processingu je opačného charakteru než modulu pre-processingu a slouží pro:

- Převod výstupních číselných hodnot na akční zásah v podobě nastavení hodnoty skutečné fyzikální veličiny. Zde se nejčastěji uplatní DA převodník, který čísla z digitální podoby převádí na hodnoty fyzikální veličiny, nejčastěji el. napětí, pomocí které je fyzický prvek

systému, např. ventil. Součástí této transformace je i tzv. *denormalizace*, která představuje přepočet z vnitřní číselné reprezentace na měřítko dané fyzikální veličiny.

– Integrace, resp. její diskrétní aproximace výstupu z fuzzy regulátoru. Podle typu fuzzy regulátoru bývá jeho výstupem buď přímo akční zásah u_t, anebo změna akčního zásahu Δu_t:

$$\Delta u_t = u_t - u_{t-1},$$

resp. diskrétní aproximace první derivace $\dot{u}$ představující navrhovanou rychlost změny akčního zásahu:

$$\dot{u} = \frac{\mathrm{d}u}{\mathrm{d}t} \approx \frac{\Delta u_t}{T}.$$

V takovém případě musíme pro získání konečné hodnoty akčního zásahu provádět průběžnou sumaci:

$$u_t = \sum_{i=1}^{t} \Delta u_i, \tag{4.5}$$

resp. aproximaci integrace:

$$u_t = \int_0^t \dot{u}\,\mathrm{d}t \approx \sum_{i=1}^{t} \Delta u_i T.$$

4.3 Typy fuzzy regulátorů

Jak bylo v předchozí části řečeno, v modulu pre-processingu se připravují různé odvozené hodnoty od aktuální odchylky vzhledem k požadované hodnotě. Obecně můžeme při sestavování pravidel znalostní báze použít libovolnou jejich kombinaci, včetně hodnot z historie regulace. Typ fuzzy regulátoru je určen tím, které hodnoty použijeme.

Rozlišujeme základní typy fuzzy regulátorů, které vycházejí z analogie klasických regulátorů. Vyjděme z ideálního spojitého klasického PID regulátoru:

$$u = K_p \left(e + \frac{1}{T_i} \int e\,\mathrm{d}t + T_d \frac{\mathrm{d}e}{\mathrm{d}t} \right). \tag{4.6}$$

Akční zásah u je dán jako lineární kombinace odchylky e, její derivace a integrálu. Parametr K_p je *proporcionální zesílení* regulátoru, T_i je *integrační časová konstanta* a T_d je *derivační časová konstanta*. Někdy vztah (4.6) zapisujeme jako:

$$u = K_p e + K_i \int e\,\mathrm{d}t + K_d \frac{\mathrm{d}e}{\mathrm{d}t}, \tag{4.7}$$

kde $K_i = \dfrac{K_p}{T_i}$ představuje *integrační zisk* a $K_d = K_p T_d$ je zesílení získané derivačním členem.

V případě digitálních regulátorů musíme vztahy aproximovat s ohledem na časovou diskretizaci danou vzorkovací periodou T. Derivační člen nahrazujeme zpětnou časovou diferencí a integrál nejčastěji sumou představující jeho obdélníkovou aproximaci[1]. Po diskretizaci tak v čase t dostaneme nejjednodušší aproximaci vztahu (4.6) ve tvaru:

$$u_t = K_p \left(e_t + \frac{1}{T_i} \sum_{j=1}^{t} e_j T + T_d \frac{e_t - e_{t-1}}{T} \right), \tag{4.8}$$

[1] Můžeme ale samozřejmě použít i přesnější aproximace, např. lichoběžníkovou nebo Simpsonovu numerickou aproximaci integrálu.

což můžeme přepsat za použití dříve definovaného značení pro hodnoty poskytované modulem pre-processingu jako:

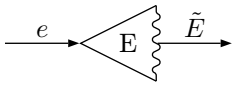
$$u_t = K_p \left(e_t + \frac{1}{T_i} \delta e_t T + T_d \frac{\Delta e_t}{T} \right), \tag{4.9}$$

resp. do tvaru:

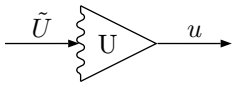
$$u_t = K_p e_t + K_i \delta e_t T + K_d \frac{\Delta e_t}{T}. \tag{4.10}$$

Jednotlivé základní typy fuzzy regulátorů jsou fuzzy aproximací speciálních případů vztahu (4.10), nebo jejich kombinací, kdy jsou vynechány některé členy, což také můžeme chápat tak, že odpovídající konstanty jsou v těchto případech nulové.

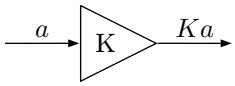
Význam symbolů v následujících blokových schématech variant fuzzy regulátorů je následující:

- 

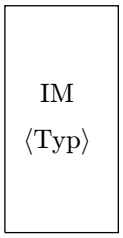
značí fuzzifikátor jazykové proměnné E, který převádí ostré hodnoty e na fuzzy množiny $\tilde{E}$. Pokud odpovídající proměnná vyjadřuje časovou dynamiku (ΔE, $\Delta^2 E$ a δE), je hodnota vstupní veličiny před fuzzifikací upravena časovou konstantou T odpovídajícím způsobem uvedeným v předchozí části při popisu modulu pre-processingu.

- 

značí defuzzifikátor jazykové proměnné U, který k fuzzy množině $\tilde{U}$ určuje její defuzzifikovanou hodnotu u. V případě defuzzifikace fuzzy množiny jazykové proměnné označující změnu akčního zásahu ΔU je defuzzifikovaná hodnota Δu ještě upravena na konečnou podobu $\frac{\Delta u_t}{T}$ z důvodů uvedených v popisu modulu post-processingu.

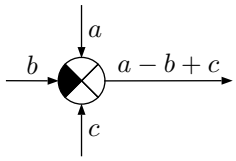
- 

představuje zesílení signálu a konstantou K na hodnotu aK.

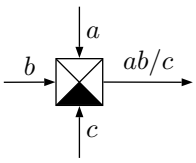
- 

zastupuje inferenční mechanismus fuzzy regulátoru odpovídajícího typu.

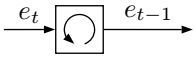
-  slouží pro sčítání hodnot, přičemž pokud je některá výseč vyplněná, znamená obrácení znaménka příslušné vstupní hodnoty, jak je uvedeno v následujícím příkladě:



-  slouží pro násobení hodnot, přičemž pokud je některý trojúhelník vyplněný, znamená převrácení hodnoty příslušné vstupní hodnoty a na hodnotu $1/a$, jak je uvedeno v následujícím příkladě:



-  funguje jako paměť předchozí hodnoty a umožňuje jednokrokové časové zpoždění vstupní hodnoty, která se na výstupu objeví až v následujícím časovém okamžiku.



4.3.1 Fuzzy P regulátor

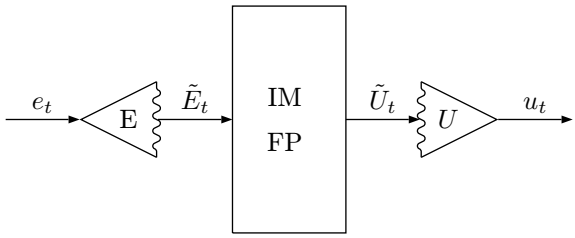
Fuzzy P regulátor je nejjednodušší variantou fuzzy regulátoru. Jedná se v podstatě o fuzzy aproximaci lineární závislosti dané analogií s klasickým P regulátorem v podobě vztahu:

$$u_t = K_p e_t. \tag{4.11}$$

Tato aproximace sama o sobě však lineární být nemusí a jak z podstaty samotné přibližné dedukce vyplývá, ani nebývá. Celkový charakter přenosové funkce je závislý na tvaru použitých fuzzy množin a především na bázi pravidel, které jsou v případě fuzzy regulátoru tohoto typu ve tvaru:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{THEN}\ U\ \mathsf{is}\ \mathscr{B}_U.$$

Tím je dána vnitřní struktura regulátoru, jejíž blokové schéma je na obr. 4.3.



Obrázek 4.3: Schéma fuzzy P regulátoru.

Pokud se budeme držet představy výše zmíněné fuzzy aproximace lineární závislosti, můžeme porovnat chování klasického P regulátoru s fuzzy P regulátorem srovnáním jejich parametrů. Uvažujme symetrické kontexty proměnných E a U, jak byly uvedeny v definici 25. Tyto udávají rozsah hodnot odpovídajících fyzikálních veličin $e \in [-E_{max}; E_{max}]$ a $u \in [-U_{max}; U_{max}]$. Z matematického hlediska se v ideálním případě jedná o lineární zobrazení:

$$u_{FP} : [-E_{max}; E_{max}] \longrightarrow [-U_{max}; U_{max}]\,,$$

které může být podle charakteru procesu buď rostoucí nebo klesající. Předpokládejme, že proces vyžaduje při kladné odchylce kladný akční zásah, z čehož máme podmínku:

$$u_{FP}(E_{max}) = U_{max},$$

ze které odvodíme obecný vztah:

$$u_{FP}(e) = \frac{U_{max}}{E_{max}} e,$$

z čehož po srovnání s (4.11) dostáváme:

$$K_p \simeq \frac{U_{max}}{E_{max}}.$$

V případě opačně reagujícího procesu bychom dostali:

$$K_p \simeq -\frac{U_{max}}{E_{max}},$$

takže můžeme psát:

$$|K_p| \simeq \frac{U_{max}}{E_{max}}. \tag{4.12}$$

Uvedený vztah (4.12), stejně jako ostatní následující srovnávací vztahy, je nutno brát pouze orientačně a spíše jako pouhou analogii, která může posloužit například pro *prvotní odhad kontextů* jazykových proměnných, pokud pro řízení procesu máme fungující klasický regulátor a chceme kvalitu regulace vylepšit jeho náhradou za fuzzy regulátor stejného typu.

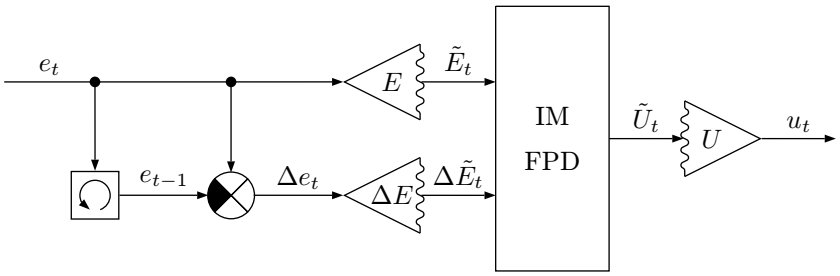
4.3.2 Fuzzy PD regulátor

Fuzzy PD regulátor je fuzzy aproximací vztahu:

$$u_t = K_p e_t + K_d \frac{\Delta e_t}{T}. \tag{4.13}$$

Jazyková pravidla mají v tomto případě tvar:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \Delta E\ \mathsf{is}\ \mathscr{A}_{\Delta E}\ \mathsf{THEN}\ U\ \mathsf{is}\ \mathscr{B}_U.$$



Obrázek 4.4: Schéma fuzzy PD regulátoru.

V prvotním hrubém přiblížení, kdy uvažujeme pouze lineární členy, můžeme odezvu regulátoru v čase t aproximovat lineárním vztahem:

$$u_t = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \Delta e_t}{T \Delta E_{max}} \right) U_{max}, \tag{4.14}$$

z čehož po srovnání s (4.13) dostaneme:

$$|K_p| \simeq \frac{U_{max}}{E_{max}}$$
$$|K_d| \simeq \frac{U_{max}}{\Delta E_{max}}.$$

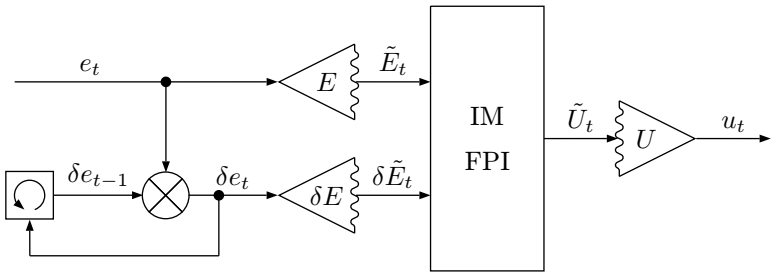
4.3.3 Fuzzy PI regulátor

Fuzzy PI regulátor je analogií ke klasickému PI regulátoru, jehož akční zásah je počítán podle:

$$u_t = K_p e_t + K_i \delta e_t T. \tag{4.15}$$

Fuzzy PI regulátor můžeme realizovat dvěma způsoby. První varianta je dána přímou aproximací vztahu (4.15) podle schématu na obr. 4.5. V tomto případě mají fuzzy IF–THEN pravidla popisující chování regulátoru tvar:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \delta E\ \mathsf{is}\ \mathscr{A}_{\delta E}\ \mathsf{THEN}\ U\ \mathsf{is}\ \mathscr{B}_U.$$



Obrázek 4.5: Schéma fuzzy PI regulátoru.

Chování regulátoru můžeme aproximovat vztahem

$$u_t = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \delta e_t}{\delta E_{max}} T \right) U_{max},$$

z čehož po srovnání s (4.15) dostaneme:

$$|K_p| \simeq \frac{U_{max}}{E_{max}}$$
$$|K_i| \simeq \frac{U_{max}}{\delta E_{max}}.$$

Inkrementální fuzzy PI regulátor

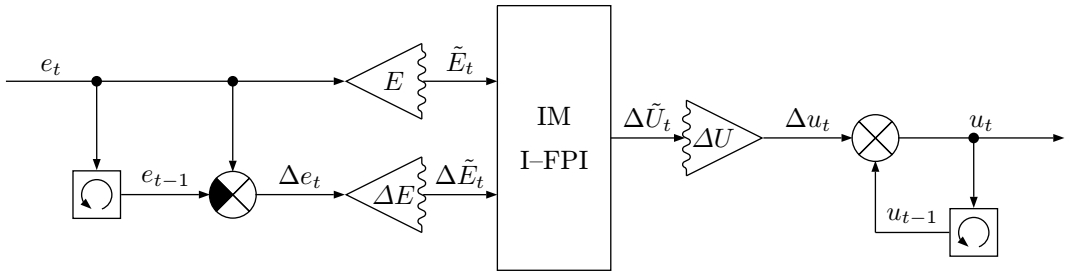
Druhou variantou fuzzy PI regulátoru je tzv. *inkrementální* fuzzy PI regulátor, jehož schéma je na obr. 4.6. Jeho název pochází z toho, že výstupem z inferenčního pravidla není přímo hodnota akčního zásahu, ale jeho změna (přírůstek) oproti stávající hodnotě. Tomu také odpovídá struktura pravidel:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \Delta E\ \mathsf{is}\ \mathscr{A}_{\Delta E}\ \mathsf{THEN}\ \Delta U\ \mathsf{is}\ \mathscr{B}_{\Delta U}.$$

Jedná se vlastně o fuzzy aproximaci vztahu odvozeného z (4.15) jeho derivací, resp. diferenciací:

$$\Delta u_t = K_p \Delta e_t + K_i e_t T.$$

Tento typ fuzzy regulátoru se v praxi používá nejčastěji, protože pravidla obsahují pouze dvě antecedentové proměnné, takže pravidla můžeme přehledně zobrazit ve dvojrozměrné tabulce, jak je uvedeno dále při řešení konkrétní úlohy regulace transformátoru tlakové energie – viz tab. (5.2). Pro člověka je navíc snažší uvažovat změnu akčního zásahu, která představuje například další pootočení řídicího ventilu oproti stávající poloze.



Obrázek 4.6: Schéma inkrementálního fuzzy PI regulátoru.

Odezvu v čase t můžeme v prvním přiblížení aproximovat vztahem:

$$\frac{\Delta u_t}{T} = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \Delta e_t}{T \Delta E_{max}}\right) \Delta U_{max}.$$

Vzhledem k tomu, že platí vztah (4.5), sumací pro u_t dostáváme:

$$u_t = \left(\frac{\pm \delta e_t}{E_{max}} T + \frac{\pm e_t}{\Delta E_{max}}\right) \Delta U_{max}, \tag{4.16}$$

takže po srovnání s (4.15) máme:

$$|K_p| \simeq \frac{\Delta U_{max}}{\Delta E_{max}}$$
$$|K_i| \simeq \frac{\Delta U_{max}}{E_{max}}.$$

4.3.4 Fuzzy PID regulátor

Fuzzy PID regulátor je nejsložitějším typem ze základních fuzzy regulátorů. Vzhledem k náročnosti jeho návrhu se používá pouze v nutných případech, kdy se jedná o regulaci velmi nelineárních a nestabilních procesů.

Uvažujeme jej v několika podobách. Struktura první varianty je vyobrazena na obr. 4.7 a vychází přímo z obecného vztahu PID regulátoru (4.10) uvedeného úvodem:

$$u_t = K_p e_t + K_i \delta e_t T + K_d \frac{\Delta e_t}{T}, \tag{4.17}$$

z čehož vyplývá i struktura pravidel ve tvaru:

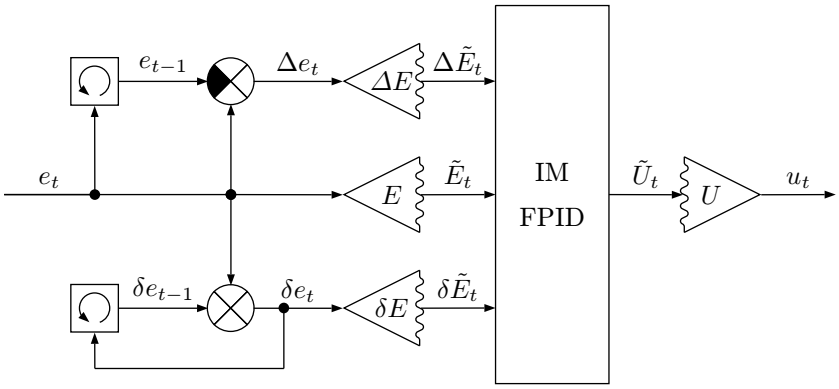
$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \Delta E\ \mathsf{is}\ \mathscr{A}_{\Delta E}\ \mathsf{AND}\ \delta E\ \mathsf{is}\ \mathscr{A}_{\delta E}\ \mathsf{THEN}\ U\ \mathsf{is}\ \mathscr{B}_U.$$

Antecedentová část pravidel je v tomto případě tvořena konjunkcí tří fuzzy výroků o hodnotách jazykových proměnných E, ΔE a δe, což je také příčinou obtížnosti jejich návrhu vzhledem k narůstajícímu počtu pravidel, kterých přibývá exponenciálně s počtem antecedentových proměnných. Přesněji, když pro každou i-tou proměnnou máme k_i jazykových výrazů, může počet pravidel narůst až do hodnoty

$$N_P = \prod_{i=1}^{n} k_i,$$

kde n je počet vstupních proměnných. Z čehož máme za předpokladu stejného počtu k jazykových výrazů pro každou jazykovou proměnnou odhad maximálního počtu pravidel ve znalostní bázi:

$$N_P = k^n.$$



Obrázek 4.7: Schéma fuzzy PID regulátoru.

Chování regulátoru lze aproximovat pomocí vztahu:

$$u_t = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \delta e_t}{\delta E_{max}} T + \frac{\pm \Delta e_t}{T \Delta E_{max}} \right) U_{max},$$

ze kterého srovnáním s (4.17) přímo vyplývá:

$$|K_p| \simeq \frac{U_{max}}{E_{max}}$$
$$|K_i| \simeq \frac{U_{max}}{\delta E_{max}}$$
$$|K_d| \simeq \frac{U_{max}}{\Delta E_{max}}.$$

Inkrementální fuzzy PID regulátor

Častěji používaným fuzzy PID regulátorem je jeho varianta v inkrementální podobě, jehož schéma je na obr. 4.8. Výstupem inferenčního mechanismu je návrh změny akčního zásahu, který se v modulu post-processingu postupně načítá pro získání konečné hodnoty akčního zásahu, obdobně jako v případě inkrementálního fuzzy PI regulátoru. Jeho výhoda oproti předchozí variantě spočívá právě v pro člověka bližšímu uvažování o změně akčního zásahu, což je důležité při sestavování jazykových pravidel regulátoru. Ta jsou v tomto případě ve tvaru:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \Delta E\ \mathsf{is}\ \mathscr{A}_{\Delta E}\ \mathsf{AND}\ \Delta^2 E\ \mathsf{is}\ \mathscr{A}_{\Delta^2 E}\ \mathsf{THEN}\ \Delta U\ \mathsf{is}\ \mathscr{B}_{\Delta U},$$

který pochází z analogie se vztahem získaným diferenciací vztahu (4.17):

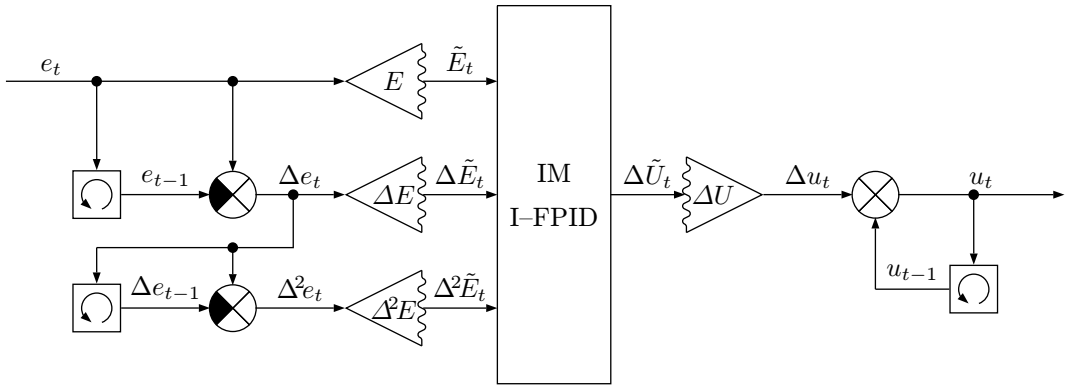
$$\Delta u_t = K_p \Delta e_t + K_i e_t T + K_d \frac{\Delta^2 e_t}{T}.$$

Chování regulátoru v hrubých rysech lze vystihnout vztahem:

$$\frac{\Delta u_t}{T} = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \Delta e_t}{T \Delta E_{max}} + \frac{\pm \Delta^2 e_t}{T^2 \Delta^2 E_{max}} \right) \Delta U_{max}.$$

Opět využijeme platnosti (4.5) a dostaneme:

$$u_t = \left(\frac{\pm \delta e_t}{E_{max}} T + \frac{\pm e_t}{\Delta E_{max}} + \frac{\pm \Delta e_t}{T \Delta^2 E_{max}} \right) \Delta U_{max},$$



Obrázek 4.8: Schéma inkrementálního fuzzy PID regulátoru.

z čehož srovnáním s (4.17) přímo vyplývá:

$$|K_p| \simeq \frac{\Delta U_{max}}{\Delta E_{max}}$$
$$|K_i| \simeq \frac{\Delta U_{max}}{E_{max}}$$
$$|K_d| \simeq \frac{\Delta U_{max}}{\Delta^2 E_{max}}.$$

Fuzzy PD+PI regulátor

Pro jednodušší vytváření báze pravidel se někdy uvažuje fuzzy PID regulátor sestrojený spojením fuzzy PD regulátoru s (inkrementálním) fuzzy PI regulátorem, jehož schéma je na obr. 4.9. Znalostní báze se v tomto případě skládá ze dvou jazykových popisů, jejichž sestavení bývá jednodušší než vytvoření jazykového popisu dvou dříve uvedených variant fuzzy PID regulátorů. To je dáno jednak tím, že je snažší při sestavování pravidel uvažovat o závislosti akčního zásahu na dvou proměnných než na třech a také celkový počet pravidel bývá obvykle menší, což je dáno i tím, že $2n^2 < n^3$ už pro $n \geq 3$. Na druhou stranu se situace trochu komplikuje tím, že výsledný akční zásah je dán součtem dvou dílčích zásahů, na což je třeba myslet při volbě jazykových kontextů jednotlivých proměnných.

Struktura pravidel jednotlivých regulátorů byla již dříve uvedena, zde ji zopakujeme kvůli označení jednotlivých jazykových proměnných. Pravidla dílčího fuzzy PD regulátoru mají tvar:

$$\mathsf{IF}\ E^{PD}\ \mathsf{is}\ \mathscr{A}_{E^{PD}}\ \mathsf{AND}\ \Delta E^{PD}\ \mathsf{is}\ \mathscr{A}_{\Delta E^{PD}}\ \mathsf{THEN}\ U^{PD}\ \mathsf{is}\ \mathscr{B}_{U^{PD}}$$

a jazykový popis dílčího fuzzy PI regulátoru se skládá z pravidel tvaru:

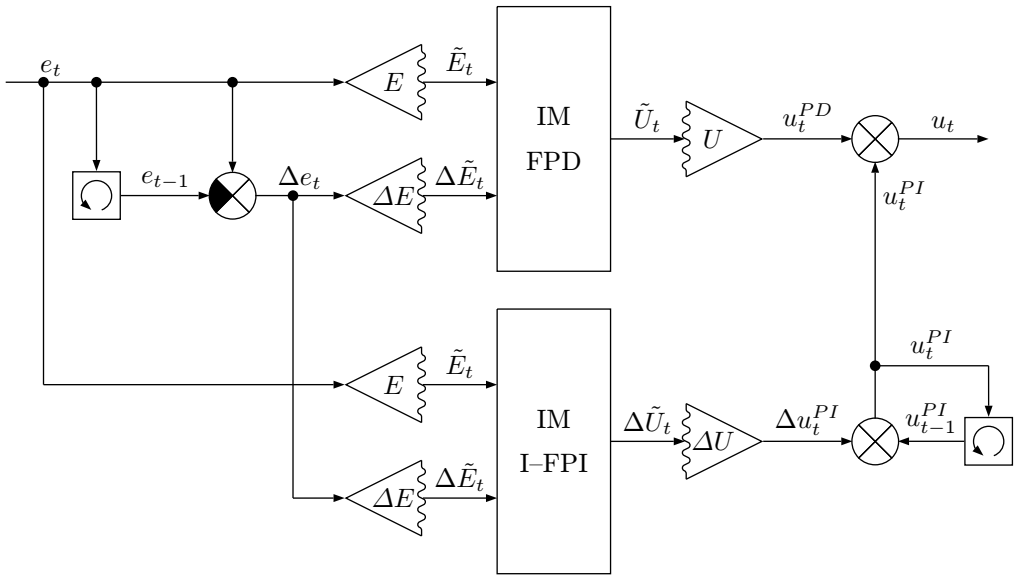
$$\mathsf{IF}\ E^{PI}\ \mathsf{is}\ \mathscr{A}_{E^{PI}}\ \mathsf{AND}\ \Delta E^{PI}\ \mathsf{is}\ \mathscr{A}_{\Delta E^{PI}}\ \mathsf{THEN}\ \Delta U^{PI}\ \mathsf{is}\ \mathscr{B}_{\Delta U^{PI}}.$$

Výsledný akční zásah je dán součtem příspěvků jednotlivých dílčích fuzzy regulátorů:

$$u_t = u_t^{PD} + u_t^{PI},$$

z čehož podle (4.14) a (4.16) máme:

$$u_t = \left(\frac{\pm e_t}{E_{max}^{PD}} + \frac{\pm \Delta e_t}{T\Delta E_{max}^{PD}}\right) U_{max}^{PD} + \left(\frac{\pm \delta e_t}{E_{max}^{PI}} T + \frac{\pm e_t}{\Delta E_{max}^{PI}}\right) \Delta U_{max}^{PI},$$



Obrázek 4.9: Schéma fuzzy PD + PI regulátoru.

tedy:

$$u_t = \left(\frac{\pm U_{max}^{PD}}{E_{max}^{PD}} + \frac{\pm \Delta U_{max}^{PI}}{\Delta E_{max}^{PI}}\right) e_t + \frac{\pm \Delta U_{max}^{PI}}{E_{max}^{PI}} \delta e_t T + \frac{\pm U_{max}^{PD}}{T \Delta E_{max}^{PD}} \Delta e_t.$$

Vzhledem k tomu, že oba dílčí regulátory jsou navrženy pro řízení stejného procesu, lze očekávat, že budou mít oba stejné znaménko u proporcionálního členu. Dostáváme se tak ke konečnému vyjádření odezvy regulátoru v čase t v podobě:

$$u_t = \pm \left(\frac{U_{max}^{PD}}{E_{max}^{PD}} + \frac{\Delta U_{max}^{PI}}{\Delta E_{max}^{PI}}\right) e_t + \frac{\pm \Delta U_{max}^{PI}}{E_{max}^{PI}} \delta e_t T + \frac{\pm U_{max}^{PD}}{T \Delta E_{max}^{PD}} \Delta e_t,$$

z čehož máme po srovnání s (4.17):

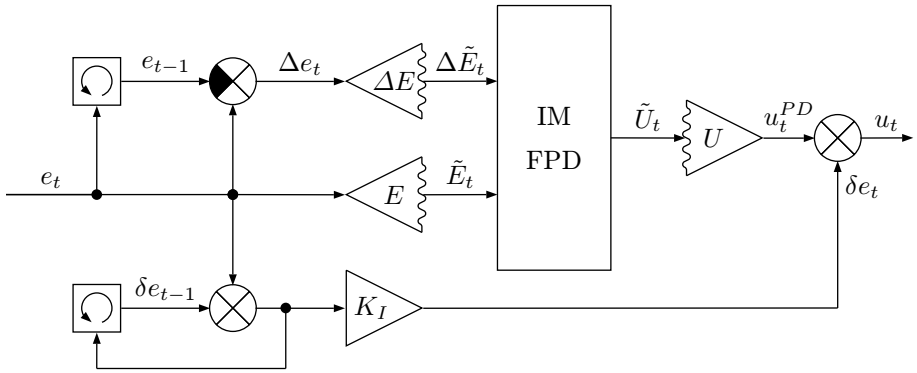
$$\begin{aligned} |K_p| &\simeq \frac{U_{max}^{PD}}{E_{max}^{PD}} + \frac{\Delta U_{max}^{PI}}{\Delta E_{max}^{PI}} \\ |K_i| &\simeq \frac{\Delta U_{max}^{PI}}{E_{max}^{PI}} \\ |K_d| &\simeq \frac{U_{max}^{PD}}{\Delta E_{max}^{PD}}. \end{aligned}$$

Fuzzy PD+I regulátor

Poslední varianta fuzzy PID regulátoru je hybridní kombinace fuzzy PD regulátoru s klasickým integračním členem, jehož schéma je na obr. 4.10.

Akční zásah v čase t můžeme v tomto případě aproximovat pomocí:

$$u_t = \left(\frac{\pm e_t}{E_{max}} + \frac{\pm \Delta e_t}{T \Delta E_{max}}\right) U_{max} + K_I \delta e_t T,$$



Obrázek 4.10: Schéma fuzzy PD+I regulátoru.

takže po srovnání s (4.17) máme:

$$|K_p| \simeq \frac{U_{max}}{E_{max}}$$
$$K_i = K_I$$
$$|K_d| \simeq \frac{U_{max}}{\Delta E_{max}}.$$

4.4 Adaptivní fuzzy regulátor

Jak už bylo v předchozí kapitole o fuzzy modelování naznačeno, fuzzy regulátor můžeme použít v roli tzv. *dohlížecího expertního systému*, kdy podle situace přepíná jednotlivé dílčí řídicí mechanismy, resp. nastavuje jejich parametry – například konstanty klasického PID regulátoru nebo hodnoty kontextů veličin jiného fuzzy regulátoru.

Zamysleme se na chvíli nad poslední zmíněnou možností. Často se v praxi stává, že se po nějakém čase regulovaná soustava ustálí ve stavu, kdy mírně osciluje okolo požadované hodnoty. Pro vylepšení regulace máme možnost buď přidat jazyková pravidla do znalostní báze, což však není možno udělat vždy, anebo se můžeme pokusit mírně změnit kontext některých jazykových proměnných. Druhá varianta má své úskalí v tom, že se změna kontextů může neblaze projevit na celkovém chování systému, kdy přestane správně reagovat v jiných stavech.

Jedním z možných řešení je právě použití dohlížecího fuzzy regulátoru, který sleduje odchylku od požadované hodnoty a ve vhodný okamžik navrhne změnu kontextů výkonného fuzzy regulátoru. Problém určení vhodného okamžiku pro změnu kontextů lze řešit různými způsoby, přičemž nejjednodušší se jeví sledování několika posledních hodnot odchylky.

Celou situaci můžeme popsat v podobě několika jednoduchých IF–THEN pravidel vystihujících danou situaci. Například: „jestliže hodnoty tří posledních odchylek byly dostatečně malé, pak zmenši kontexty všech veličin na polovinu“. Tímto se z malých odchylek stanou pro výkonný regulátor odchylky velké. Jazyková pravidla tedy mohou mít například následující strukturu:

$$\mathsf{IF}\ E_t\ \mathsf{is}\ \mathscr{A}_{E_t}\ \mathsf{AND\ IF}\ E_{t-1}\ \mathsf{is}\ \mathscr{A}_{E_{t-1}}\ \mathsf{AND\ IF}\ E_{t-2}\ \mathsf{is}\ \mathscr{A}_{E_{t-2}}\ \mathsf{THEN}\ K\ \mathsf{is}\ \mathscr{B}_K,$$

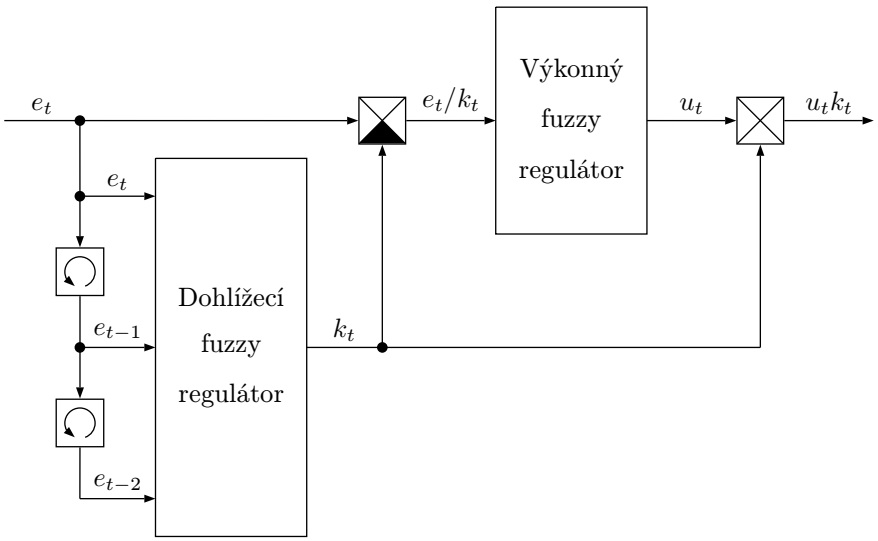
kde výstupní jazyková proměnná K představuje měřítko v podobě koeficientu, kterým se násobí hodnoty kontextů výkonného fuzzy regulátoru. Výše uvedený příklad můžeme tedy přepsat např.

do podoby:

$$\text{IF } E_t \text{ is } \pm \textit{ve sm} \text{ AND IF } E_{t-1} \text{ is } \pm \textit{ve sm} \text{ AND IF } E_{t-2} \text{ is } \pm \textit{ve sm} \text{ THEN } K \text{ is } \textit{sm}.$$

Tento přístup dává také přímočarou možnost ke zpětnému návratu k původním hodnotám, pokud z jakýchkoli příčin dojde k nárůstu odchylky od požadované hodnoty. Stačí k tomuto účelu vytvořit odpovídající jazyková pravidla a přidat je do báze pravidel dohlížecího regulátoru.

Nejjednodušší variantu struktury adaptivního regulátoru s dohlížecím fuzzy regulátorem je možno vidět na obr. 4.11. Fuzzifikační a defuzzifikační členy, stejně jako pre-procesingový modul připravující potřebné odvozené hodnoty od odchylky e_t jsou pro jednoduchost uvažovány jako součást bloků jednotlivých regulátorů. Aplikace násobícího faktoru k_t na kontexty jazykových proměnných výkonného fuzzy regulátoru je pro názornost realizována v podobě přímé modifikace hodnot veličin e_t a u_t. V případě vstupních proměnných to znamená dělení vstupní veličiny hodnotou k_t a u výstupní proměnné spočívá změna kontextu v násobení akčního zásahu hodnotou k_t.



Obrázek 4.11: Schéma adaptivníno regulátoru s dohlížecím fuzzy regulátorem.

V myšlenkové úvaze můžeme jít ještě dále celý postup aplikovat i na samotný dohlížecí fuzzy regulátor. To znamená, že až se regulovaná soustava opět dostatečně přiblíží požadované hodnotě, přičemž nyní uvažujeme o hodnotách k-násobně menších, můžeme opět zmenšit kontexty uvažovaných proměnných.

Této opakované změny kontextů dosáhneme modifikací struktury zapojení, kdy vstupem dohlížecího fuzzy regulátoru je hodnota e_t/k_{t-1}, čímž se pro něj vytvoří uzavřená zpětnovazební smyčka, jak je znázorněno na obr. 4.12. Aby nedocházelo k okamžitému zpětnému zvětšení kontextů a následné oscilaci hodnoty násobícího faktoru k, je nyní výstupem dohlížecího fuzzy regulátoru průběžná změna ∇k_t, která se postupně kumuluje pomocí vztahu:

$$k_t = \prod_{i=1}^{t} \nabla k_i.$$

Hodnota $\nabla k_t = 1$ tak představuje zachování násobícího faktoru k na stávající hodnotě. Kontext jazykové proměnné ∇K tak může být například $\left\langle \frac{1}{2}; 1; 2 \right\rangle$. Regulační strategii dohlížecího fuzzy

regulátoru můžeme velmi zhruba nastínit pomocí pravidel:

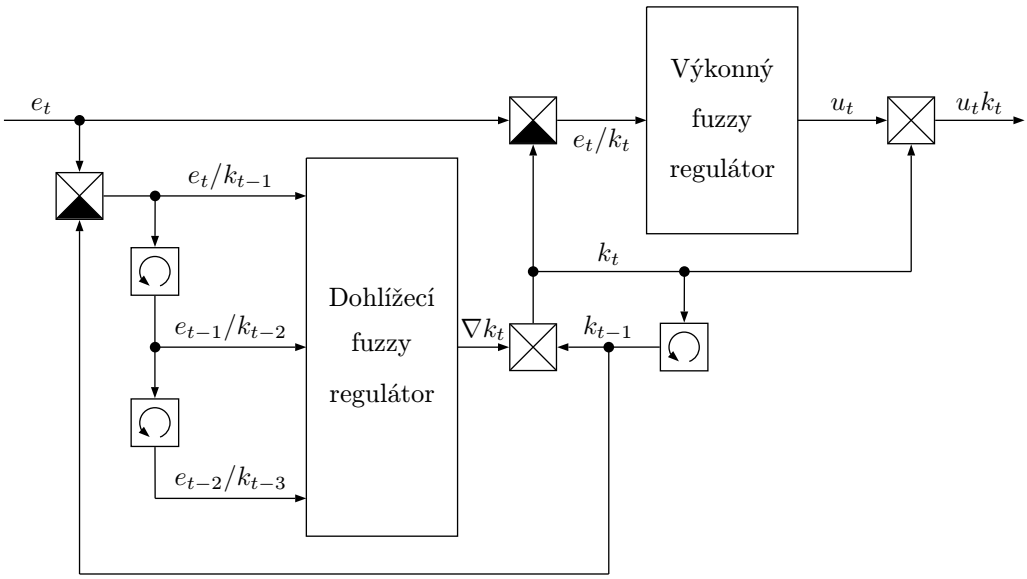
IF E_t is $\pm$ *ve sm* AND IF E_{t-1} is $\pm$ *ve sm* AND IF E_{t-2} is $\pm$ *ve sm* THEN ∇K is *sm*

IF E_t is $\pm$ *me* AND IF E_{t-1} is $\pm$ *me* AND IF E_{t-2} is $\pm$ *me* THEN ∇K is *me*

IF E_t is $\pm$ *bi* AND IF E_{t-1} is *ignored* AND IF E_{t-2} is *ignored* THEN ∇K is *bi*

$\vdots$

Třetí uvedené pravidlo popisuje situaci, kdy došlo k přílišnému zvýšení odchylky od požadované hodnoty a je zapotřebí naopak kontexty veličin vrátit k původním vyšším hodnotám. To lze udělat buď opět postupným zvyšováním pomocí hodnoty násobícího koeficientu $\nabla k_t > 1$, avšak v některých případech, obzvláště při reakci na změnu řídicí veličiny udávající požadovaný stav, je nutné udělat rázný zásah v podobě nastavení hodnot kontextů na počáteční hodnoty, což lze provést přímým nastavením hodnoty k_t na hodnotu 1.



Obrázek 4.12: Schéma adaptivníno regulátoru s dohlížecím fuzzy regulátorem uzavřeným ve zpětnovazební smyčce.

Výhodou při použití adaptivních fuzzy regulátorů je, že pro převedení regulované soustavy do stavu velmi blízkého požadovanému stavu, stačí mnohdy poměrně hrubý jazykový popis regulační strategie.

Kapitola 5

Praktická aplikace fuzzy regulace: Transformátor tlakové energie na bázi minerální olej – voda

Jedním z hlavních uskutečněných praktických výsledků je úspěšné použití fuzzy regulátoru při řešení projektu FD-K3/033 v rámci programu projektového konsorcia s názvem „Výzkum a vývoj zařízení pro transformaci přenosu tlakové energie na bázi minerální olej – voda a realizace technologie řízení tohoto procesu“. Projekt se uskutečnil v letech 2003 – 2005 ve spolupráci s hlavním řešitelem projektu, kterým byla firma KOEXPRO OSTRAVA, a. s. Naše účast na řešení spočívala především v realizaci regulace řízení výsledného zařízení.

Celkem se na řešení tohoto projektu podílelo 5 řešitelů. Jeden hlavní řešitel, kterým byla již dříve zmíněná společnost KOEXPRO (ředitel Ing. Jaroslav Pařenica) a 4 spoluřešitelé. Jmenovitě to byli:

- INTERFLUID spol. s r.o. (Doc. Ing. Václav Sivák, CSc.)
- BIC Ostrava s.r.o. (Ing. Jaroslav Kunčický, CSc.)
- JANYTOM (Jiří Janeczko)
- Ostravská univerzita v Ostravě / Centrum informačních technologií (RNDr. Martin Malčík, Ph.D., prof. Ing. Vilém Novák, DrSc., Mgr. Viktor Pavliska)

5.1 Popis problematiky

Transformace přenosu tlakové energie mezi dvěmi různými nositeli pramení z požadavků praxe a nutnosti nalézt nové, netradiční způsoby využívání tlakové energie, kde na jedné straně je nositelem energie ekologicky nezávadná a nehořlavá kapalina a na druhé straně, v uzavřeném prostředí, minerální olej nebo emulze. K tomu je však zapotřebí oddělit přenosová média prvkem – *transformátorem*, který umožní využívat hydraulických prvků konstruovaných na minerální olej spolu s tlakovým zdrojem na emulzi nebo jiné médium.

Základní myšlenka projektu spočívá v tom, že pro dvě různá prostředí se budou využívat v rámci jednoho subsystému přenosu energie dva různí nositelé s tím, že se mezi ně vloží zařízení pro transformaci tlakové energie pracující s oběma různými nositeli. Výhodou navrhovaného řešení projektu je, že přenos tlakové energie může být rovněž *inverzního* charakteru, přičemž vlastní zařízení pro transformaci mezi dvěma prostředími je univerzálního typu a není nutno je upravovat dle směru transformace.

5.2 Vývoj zařízení pro transformaci přenosu energie

Teorie transformace energie je postavena na paralelním zařazení transformátoru v obvodu s dalšími spotřebiči. Transformátor odebírá z obvodu pouze takový výkon, jaký spotřebovává připojený spotřebič. V obvodu transformátoru tedy nedochází ke škrcení a maření energie. Kritéria, která musí splňovat zařízení pro transformaci přenosu tlakové energie na bázi minerální olej – voda, jsou dána použitím tohoto zařízení.

Schématicky můžeme zapojení transformátoru do obvodu s rozvodem kapaliny 1 zobrazit na obrázku 5.1, přičemž význam jednotlivých veličin vyskytujících se ve schématu je následující:

P_N – nevyužitý výkon

P_1 – příkon – výkon v obvodu kapaliny 1

P_2 – výkon v obvodu kapaliny 2

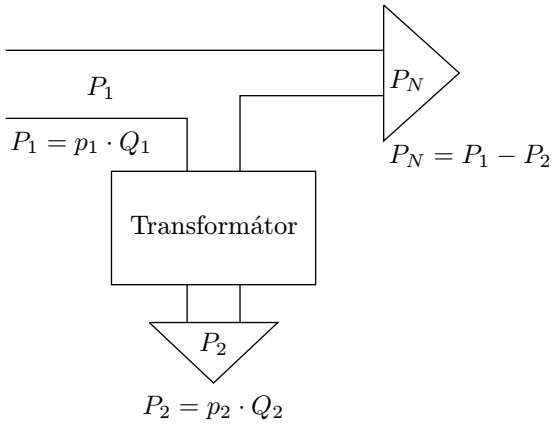
η_Q – průtoková účinnost transformátoru

η_{mp} – mechanicko-tlaková účinnost transformátoru

P_{NS} – stálý nevyužitý výkon – vzniklý vlivem doby přestavení rozvaděče

a jsou vzájemně svázány vztahem:

$$P_2 = P_1 \cdot \eta_Q \cdot \eta_{mp} - P_{NS}.$$



Obrázek 5.1: Schéma přenosu energie transformátorem.

Pro řízení procesu transformace byly teoreticky rozpracovány varianty regulace, odvíjející se od volby řídící veličiny. Podle druhu aplikace může být regulátor buď univerzální se třemi nastavitelnými módy, nebo pro konkrétní aplikace v jednom ze tří provedení. Uvažovány byly tyto možné varianty:

- Regulace na konstantní *tlak*
  Regulátor udržuje konstantní nastavený tlak p_s a podle odebíraného průtoku Q_2 zmenšuje nebo zvětšuje prodlevu mezi zdvihy pístu t_0. Při zvýšení odebíraného průtoku regulátor zajistí snížení prodlevy mezi zdvihy až do dosažení nastaveného tlaku p_s a naopak.

- Regulace na konstantní *průtok*
  Při zvýšení odporu zátěže Δp_z dojde ke snížení průtoku Q_2. Regulátor na základě informace o snížení průtoku Q_2 sníží prodlevu mezi zdvihy t_0 a tím se zvýší množství kapaliny v akumulátoru a tím i tlak p_2, dojde k překonání odporu zátěže Δp_z a ke zvýšení průtoku na požadovanou hodnotu.

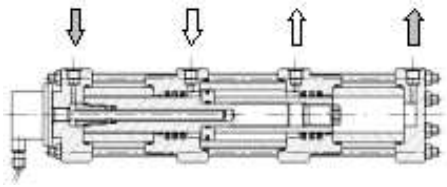
- Regulace na konstantní *výkon*
  Regulátor má naprogramovanou funkci: závislost tlaku na průtoku: $P = p \cdot Q$. Při snížení nebo zvýšení průtoku Q_2 regulátor podle této funkce pomocí snížení, nebo zvýšení prodlevy mezi zdvihy t_0 nastaví množství kapaliny v akumulátoru a tím tlak p_2. Minimální tlak je však dán typem a velikostí akumulátoru.

5.3 Konstrukční charakteristika zařízení

Rozborem kritérií optimálního výběru variant bylo rozhodnuto o realizaci čtyř variant prototypů přenosu tlakové energie, přičemž pátá varianta (RIT) byla doplněna v průběhu řešení projektu. Jmenovitě se jedná o:

Dvojčinný inverzní transformátor – DIT

Základním prvkem transformátoru je dvojčinný píst konající vratný pohyb, který odděluje prostory pracovních kapalin mezi primární a sekundární stranou. Pomocí změny poměru mezi primární a sekundární plochou pístu je možno jej vyrobit i v provedení pro multiplikaci tlaku.

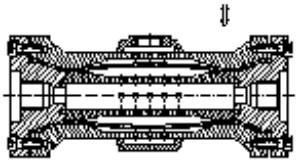


Membránový inverzní transformátor – MIT

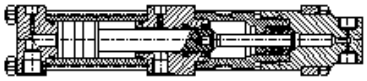
Membránový (vakový) inverzní transformátor MIT je vytvořen malou konstrukční úpravou z hydraulického akumulátoru, kde původní plynová strana byla přizpůsobena pro hydraulický olej. Dvě různé pracovní kapaliny odděluje tenká pružná membrána, která umožňuje transformaci přenosu tlakové energie s vysokou účinností.

Pryžový inverzní transformátor – QIT

Ústředním prvkem transformátoru je pryžová hadice, která odděluje pracovní kapaliny a zároveň plní funkci těsnění. Transformace přenosu tlakové energie je založena na opakované deformaci pryžové hadice.

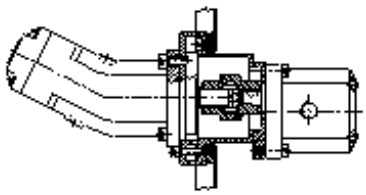


Pístový inverzní transformátor – PIT



Pístový inverzní transformátor PIT je vyvinut jako svorníková tlaková nádoba. Prostory pracovních kapalin mezi primární a sekundární stranou odděluje píst. Pokud je nutné zajistit nemísení obou pracovních kapalin, musí být píst těsněn.

Rotační inverzní transformátor – RIT



Rotační inverzní transformátor RIT je konstrukční uzel tvořený hydromotorem a hydrogenerátorem se spojenými hřídeli. Transformátor RIT byl vyvinut jako funkční vzor pro nasazení do prostředí s nebezpečím výbuchu hořlavých plynů, par a prachu. Proto byl maximálně zjednodušen bez možnosti elektronické regulace.

Všechny uvedené varianty prototypů byly vyvíjeny a testovány současně, aby se ukázalo, které budou nejlépe splňovat stanovená kritéria. Pro podrobnější popis daných prototypů viz [15].

Pro zlepšení průběhu tlaku a průtoku bylo v průběhu řešení rozhodnuto o současném použití tří transformátorů fázově řízených se zpožděním 120°. V principu to odpovídá třípístovému čerpadlu s klikovým mechanismem ze strany pohonu.

Pomocí fuzzy regulace byla řízena varianta transformátoru PIT, QIT a DIT – a to pomocí řízení přepínání elektrohydraulických rozvaděčů regulováním frekvence přestavování. S ohledem na požadovaný tlak, který byl snímán pomocí tlakového čidla ve vyrovnávacím akumulátoru, byla regulována délka impulsů $T_i = 1/f$.

Všechny typy transformátorů mohou pracovat v inverzním zapojení, což znamená přenos tlakové energie směrem minerální olej – kapaliny HFA, HFC nebo opačně.

5.4 Postup algoritmického řešení fuzzy regulace

Vzhledem k tomu, že na začátku řešení projektu neexistovalo žádné zařízení, na kterém by bylo možno testovat a zkoušet funkci fuzzy regulátoru, aby práce týkající se vývoje fuzzy regulace nestála a nemuselo se čekat až na první funkční prototyp, byl firmou JANYTOM vytvořen elektronický simulátor daného zařízení. Jedná se o fyzické zařízení na bázi PLC. Toto zařízení bylo vytvořeno tak, aby na něm bylo možno ladit fuzzy regulátor. Základním předpokladem ovšem bylo, že díky robustnosti fuzzy regulátoru při jeho nasazení na skutečný funkční prototyp transformátoru nebude zapotřebí provést zásadní změny v jazykovém popisu, což se také nakonec potvrdilo.

Celý postup řešení projektu z hlediska fuzzy regulace je možno rozdělit do několika dílčích etap:

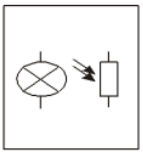
1. Určení veličin vystupujících v regulované soustavě
2. Navržení typu fuzzy regulátoru
3. Sestrojení PC verze fuzzy regulátoru
4. Sestavení báze pravidel a její odladění na simulátoru
5. Testování fuzzy regulátoru na funkčním prototypu transformátoru

6. Převod řídící logiky regulátoru z PC do PLC pro konečnou verzi

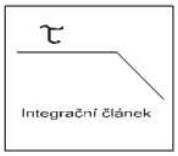
Simulátor transformátoru

Pro potřeby simulace a odladění byl vybrán průmyslový automat PLC fy ABB 07KT51/24VDC doplněný analogovým modulem XE08B5. Pro konečné řízení procesu pak byly v samotném závěru projektu konečné odladěné algoritmy fuzzy řízení převedeny do jednočipového mikrokontroléru, jehož výhody spočívají především v nižší pořizovací ceně, v menších rozměrech a v neposlední řadě má nižší spotřebu el. energie, což je asi nejdůležitější výhoda v případě, kdy je elektronika celého řízení napájena z přenosné baterie.

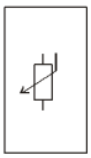
Elektronické schéma zapojení fuzzy regulátoru se simulátorem děje X-BOX je v blokové struktuře znázorněno na obrázku A.9, přičemž vlastní fuzzy regulátor je v této počáteční variantě implementován na straně PC. Samotná struktura modulu X-BOX je detailněji zobrazena na obrázku 5.2, přičemž význam jednotlivých symbolů je následující:



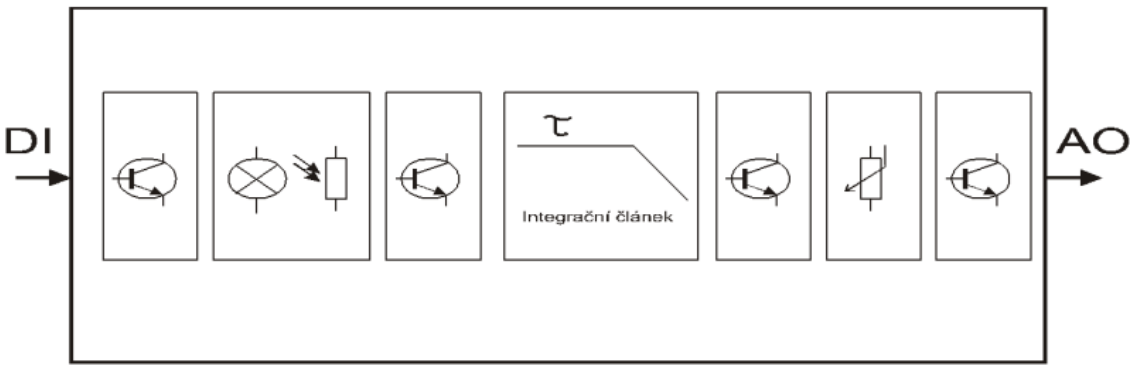
Představuje blok zajišťující simulaci zpoždění tlakových spínačů, který byl realizován pomocí žárovky, která svítila na fotorezistor. Vlastní zpoždění je dáno fyzikálními vlastnostmi vlákna žárovky, u něhož dochází ke zpoždění svítivosti od okamžiku zapnutí až do dosažení maximálního jasu.



Integrační článek slouží k simulaci akumulační schopnosti olejového akumulátoru. Vlastní elektronické provedení sestávalo z kondenzátoru o dostatečné kapacitě, jehož fyzikální vlastnosti umožňují poměrně věrně simulovat akumulaci el. energie, obdobně jako olejový akumulátor je schopen pracovat s tlakovou energií.



Tento blok slouží k simulaci různých odběrů tlakového média a byl realizován pomocí otočného rezistoru s nastavitelnou velikostí odporu.



Obrázek 5.2: Blokové schéma modulu simulátoru X-BOX.

V závěrečných fázích projektu, kdy už existovaly funkční prototypy a bylo možno testovat vlastnosti použitého fuzzy regulátoru, byl simulátor rozšířen o schopnost ovládání přepínacích

ventilů. Schéma zapojení pro řízení funkčních prototypů je možno vidět na obrázku A.10. Dále jej pak bylo možno používat jednak jako simulátor, bez nutnosti napojení na hydraulický obvod, a po přepnutí do řídícího režimu bylo možno použít stejné zařízení pro přímé řízení prototypu transformátoru tlakové energie. Tato možnost využití jednoho zařízení pro dva účely měla výhodu zejména v tom, že nebylo zapotřebí měnit komunikační protokol mezi PC a PLC.

Typ použitého fuzzy regulátoru

Pro účely projektu byl zvolen fuzzy regulátor typu PI v inkrementální variantě. Jazyková báze je tedy sestavena z pravidel ve tvaru:

$$\mathsf{IF}\ E\ \mathsf{is}\ \mathscr{A}_E\ \mathsf{AND}\ \Delta E\ \mathsf{is}\ \mathscr{A}_{\Delta E}\ \mathsf{THEN}\ \Delta U\ \mathsf{is}\ \mathscr{B}_{\Delta U}.$$

Tento fuzzy regulátor byl nejprve implementován na PC, komunikující s již dříve zmíněným PLC obvodem po sériovém portu, a posléze v závěru řešení projektu byl ve formě řídící tabulky převeden do mikrokontroléru. Z hlediska struktury řízení je regulátor zapojen do obvyklé zpětnovazební regulační smyčky, uvedené v části 4.1 na obr. 4.1.

Funkční vzor fuzzy regulátoru

Vlastní fuzzy regulace probíhá v PC a PLC je prostředníkem, který po sériové sběrnici RS-232 komunikuje s PC. To znamená, že převádí analogová data z čidla tlaku na digitální a posílá tato data do PC a zároveň přijímá data z PC ve formě hodnoty a převádí je na periodu spínání jednotlivých ventilů.

Na straně PC se praktická realizace fuzzy regulátoru skládá ze dvou hlavních částí:

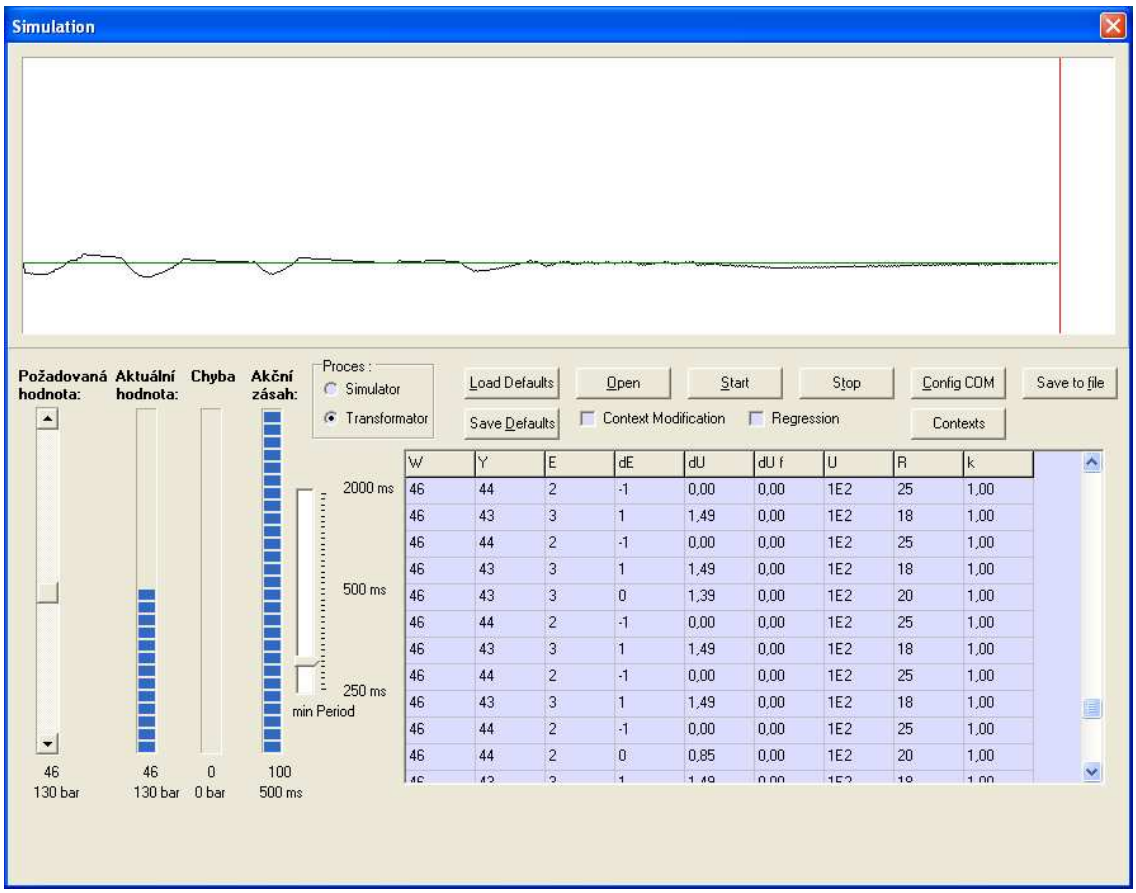
1. *Řídící jádro*
   Toto jádro představuje vlastní fuzzy inferenční mechanismus, který zajišťuje odpovídající odezvy na aktuální stav řízeného procesu. Z implementačního hlediska se jedná o COM objekt ze softwarového balíku LFLC 2000, který tvoří komunikační rozhraní mezi knihovnou `IRAFMlib` (obsahující implementaci rutin potřebných pro fuzzy inferenci) a ostatními aplikacemi pro OS Windows.

2. *Zpětnovazební smyčka*
   Do této smyčky je umístěno výše zmíněné řídící jádro. Cílem tohoto celku je zajišťovat příslušné údaje pro rozhodovací systém z řízeného procesu a naopak aplikovat získané akční zásahy z fuzzy regulátoru do řízeného procesu.

Vlastní zpětnovazební smyčka byla implementována jako součást aplikace sloužící pro účely simulace a ladění jazykového popisu. Tato aplikace zprostředkovávala rovněž komunikaci se zařízením řídícím regulovaný systém. Celá aplikace je napsána ve vývojovém nástroji Borland C++ Builder verze 5.0 a její vzhled ukazuje obrázek 5.3. Pro vytvoření pravidel fuzzy regulátoru a jejich následnou editaci při ladění byl použit softwarový balík LFLC 2000.

Báze pravidel

Hodnoty kontextů jednotlivých veličin byly na základě prvotního odhadu a následného ladění v konečné podobě pro simulátor stanoveny takto:



Obrázek 5.3: Snímek obrazovky aplikace realizující fuzzy regulátor na PC

	min.	max.
E	-80	80
ΔE	-7	7
ΔU	-1.5	1.5

Báze znalostí se může obecně skládat ze soustavy jazykových popisů. V našem případě jsme vystačili s jedním jazykovým popisem, jehož úplný výpis je v tabulce 5.1. Tento jazykový popis můžeme vzhledem k počtu vstupních proměnných zobrazit také v dvojrozměrné podobě, jak je vidět v tabulce 5.2. Díky robustnosti fuzzy regulace nebylo při přechodu ze simulátoru na prototyp transformátoru zapotřebí do báze pravidel vůbec zasahovat. Jediná nutná změna báze dat se týkala hodnot kontextů jednotlivých veličin.

5.5 Testování regulátoru na funkčním prototypu

5.5.1 Rozdíly v regulaci simulátoru a prototypu transformátoru

Mezi regulací simulátoru a vytvořených prototypů tlakové energie je několik rozdílů, které bylo zapotřebí zohlednit při realizaci fuzzy regulátoru. Jedná se především o interpretaci vstupních

a výstupních veličin. Nyní následuje stručný popis těchto rozdílů a provedených změn v řídícím jádře fuzzy regulátoru.

Nejdůležitějším rozdílem oproti regulaci simulátoru je interpretace akčního zásahu fuzzy regulátoru. V obou případech může akční zásah nabývat hodnot z intervalu $[0; 100]$.

- V případě simulátoru je interpretace ve formě údaje představujícího procentuální vyjádření toho, jakou část z předem napevno určené délky periody bude svítit žárovka na fotorezistor, přičemž hodnota $u = 0$ znamená, že žárovka vůbec nesvítí, například hodnota $u = 50$ představuje stav, kdy žárovka svítí polovinu periody a hodnota $u = 100$ je zadána v případě, že má žárovka svítit celou dobu periody.

- V případě prototypu transformátoru představuje hodnota akčního zásahu délku periody jednoho cyklu, ve kterém dochází k přepínání jednotlivých hydraulických ventilů. Hodnota $u = 0$ znamená, že se má cyklus přepínání ventilů zastavit. Pro hodnoty různé od 0 platí následující vztah:
$$T = u \cdot 50ms, \tag{5.1}$$
kde T je nastavovaná délka periody přepínacího cyklu.

Tento rozdíl má největší dopad na celkový průběh regulace, protože „obrací“ chování procesu na akční zásah co se týče zvyšování a snižování aktuální hodnoty regulované veličiny.

- V případě simulátoru se zvýšení regulované hodnoty dosáhlo pomocí zvýšení akčního zásahu (poměrné délky osvitu fotorezistoru).

- V případě prototypu transformátoru se zvýšení hodnoty regulované veličiny dosáhne snížením hodnoty akčního zásahu (zkrácením periody přepínání ventilů).

Pro realizaci tohoto rozdílu je nutno v zájmu zachování původního způsobu regulace transformovat původní hodnotu akčního zásahu na hodnotu příslušející k procesu řízení prototypu transformátoru. Pro jednoduchost byla zvolena lineární transformace ve tvaru

$$u' = a \cdot u + b,$$

kde hodnoty konstant byly určeny tak, aby splňovaly následující podmínky:

1. Původní hodnotě $u = 100$ odpovídá hodnota $u' = T_{min}$, která představuje nejmenší možnou hodnotu délky periody jednoho cyklu vycházející z reálných fyzických možností přepínacích ventilů. Pro jeden ventil je tato doba po přepočtení podle vztahu (5.1) přibližně $400ms$. Pro cyklus obsahující přepnutí dvou ventilů je tato doba dvojnásobná, tedy přibližně 800ms. V upravené aplikaci je možno tuto hodnotu interaktivně nastavovat pro testovací účely.

2. Původní hodnotě $u = 0$ odpovídá hodnota $u' = 100$, která po přepočtu na milisekundy odpovídá nastavení periody na hodnotu 5 000 ms, tedy 5s, což téměř odpovídá praktickému zastavení.

Těmto podmínkám vyhovují hodnoty konstant vypočtené dle následujících vztahů:

$$\begin{aligned} a &= \frac{T_{min}}{100} - 1 \\ b &= 100. \end{aligned}$$

Co se týče interpretace vstupní veličiny, která je zároveň regulovanou veličinou, je situace jednodušší, protože její odlišná interpretace nemá zásadní vliv na samotný průběh regulace. V obou případech může regulovaná veličina nabývat hodnot z intervalu $y \in [0; 100]$.

- V případě simulátoru zařízení představuje získaná hodnota velikost napětí na kondenzátoru ve voltech vynásobenou hodnotou 100.
- V případě prototypu transformátoru se jedná o hodnotu naměřenou na tlakovém čidle z řídícího obvodu a její interpretace je důležitá zejména pro zobrazovací účely a snadnější nastavování požadované hodnoty w, která z přirozeného důvodu může nabývat hodnot ze stejného intervalu jako regulovaná veličina.

Přepočet naměřené hodnoty z tlakového čidla na tlak v jednotkách bar je závislý na typu použitého čidla. V našich podmínkách bylo použito čidlo s téměř lineární charakteristikou, takže transformační vztah je ve tvaru:

$$P = a_p \cdot y + b_p,$$

kde hodnoty konstant a_p, b_p se určí tak, aby splňovaly následující podmínky:

1. Naměřené hodnotě $y = 20$ odpovídá tlak 0 bar.
2. Naměřené hodnotě $y = 100$ odpovídá tlak P_{max}, který je dán charakteristikou daného tlakového čidla. V našem případě bylo použito čidlo s hodnotou $P_{max} = 400$ bar.

Těmto podmínkám vyhovují hodnoty konstant vypočtené dle následujících vztahů:

$$\begin{aligned} a_p &= \frac{P_{max}}{80} = 5 \\ b_p &= -\frac{P_{max}}{4} = -100. \end{aligned}$$

Vzhledem k výše zmíněným odlišnostem byla aplikace realizující fuzzy regulaci na straně PC přizpůsobena tak, aby mohla sloužit k regulaci obou zařízení. Toto je zajištěno pomocí přepínače, který z uživatelského rozhraní aplikace umožňuje volbu odpovídající varianty.

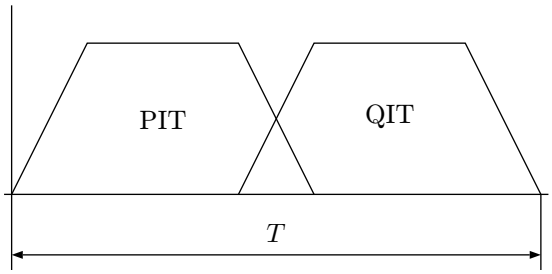
5.5.2 Měření a testování fuzzy regulace na prototypech

Měření a testování fuzzy regulátoru bylo prováděno na prototypech transformátorů tlakové energie typu PIT, QIT a DIT.

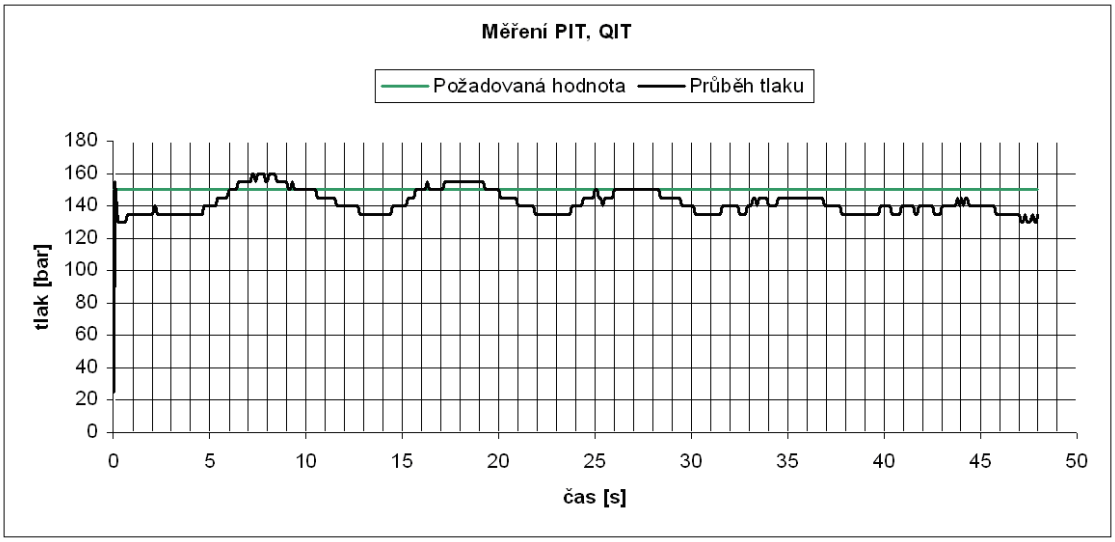
Transformátory PIT a QIT

Transformátory PIT a QIT byly spojeny do jednoho obvodu a byly testovány souběžně. Jednoduše lze průběh přepínání těchto dvou transformátorů pomocí hydraulických ventilů znázornit obrázkem 5.4, ve kterém T představuje již dříve zmiňovanou délku periody jednoho cyklu, kterou fuzzy regulátor nastavuje jako akční zásah pro získání požadovaného chování transformátoru.

Graf na obrázku 5.5 ukazuje závěrečné testovací měření, při kterém byla nastavena požadovaná hodnota tlaku na 150 bar.



Obrázek 5.4: Časový průběh přepínání PIT a QIT transformátorů.



Obrázek 5.5: Průběh tlaku na spřažených transformátorech PIT a QIT.

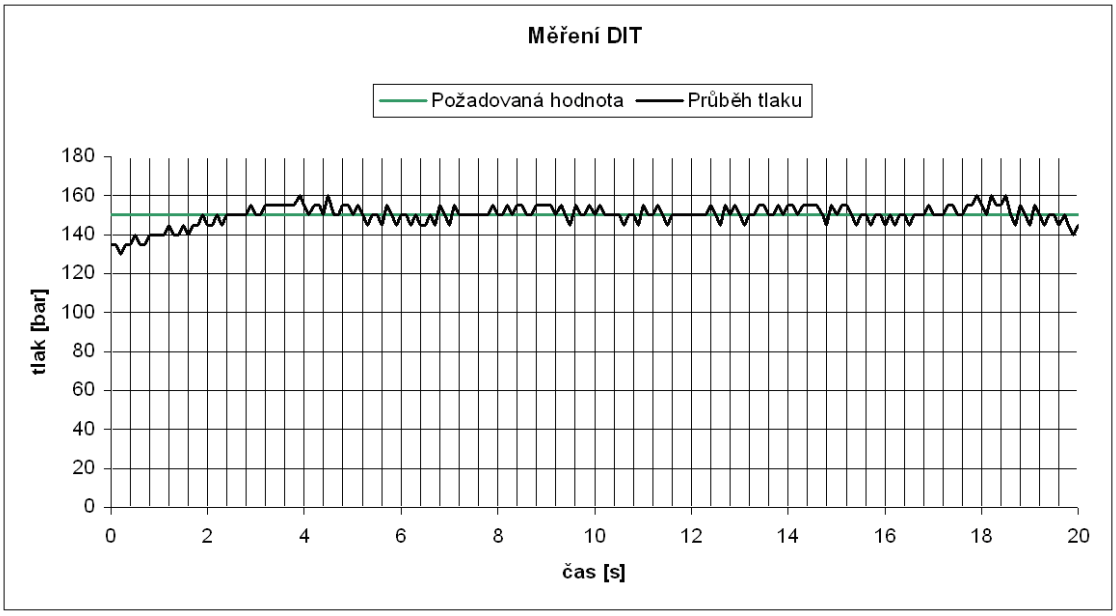
Nejdůležitějším výsledkem testovacích měření bylo určení kontextů vstupních a výstupních veličin, které určuje odpovídající průběh tlaku z předchozího grafu. Nejlepších výsledků bylo dosaženo při hodnotách:

	min.	max.
E	-15	15
ΔE	-7	7
ΔU	-16	16

Transformátor DIT

Pro transformátor DIT byly vytvořeny stejné testovací podmínky jako pro transformátory PIT a QIT. Počáteční hodnoty kontextů byly nastaveny rovněž na hodnoty získané z měření PIT a QIT. Průběh tlaku při regulaci transformátoru PIT je vidět na grafu z obrázku 5.6.

Transformátor DIT byl dále testován s různými typy zátěží, aby se prověřilo chování fuzzy regulátoru při různých charakteristikách odběru. Jako první byl do obvodu zapojen hydraulický



Obrázek 5.6: Průběh tlaku na transformátoru DIT.

utahovák, který měl enormní nároky na průtok dodávaný transformátorem. Tyto nároky se rovnaly maximálnímu průtoku, který byl transformátor schopen dodávat. S výše nastavenými kontexty veličin fuzzy regulátor nestačil dostatečně rychle zareagovat na požadovaný odběr, což se projevilo náhlým poklesem tlaku, který se již při stálém maximálním odběru nepodařilo vyrovnat na požadovanou hodnotu.

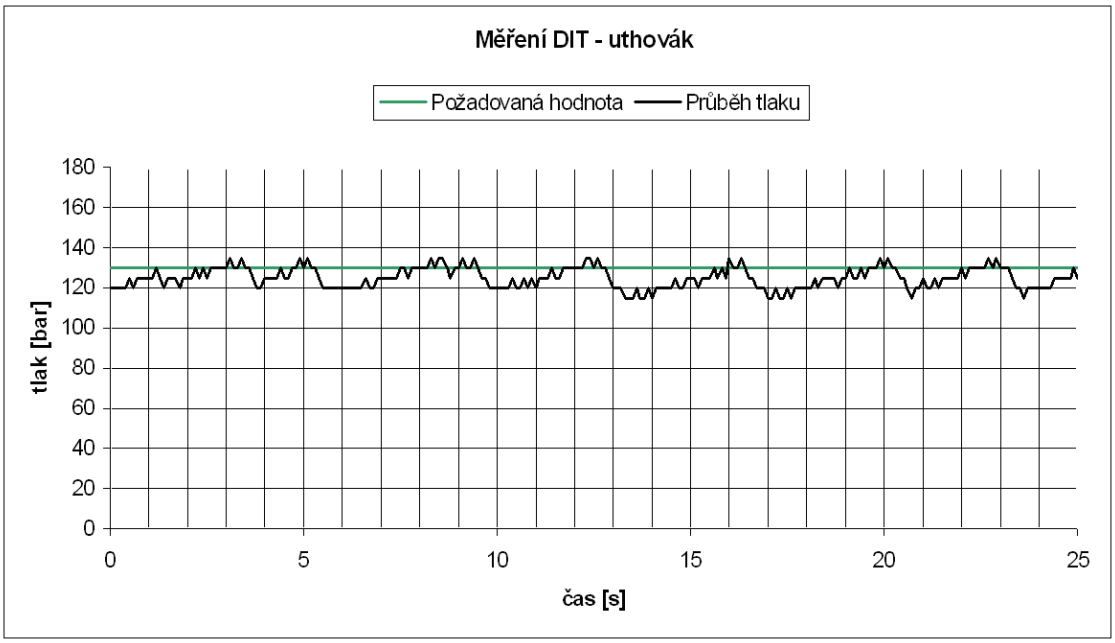
Pro uspokojení těchto požadavků byly parametry kontextů veličin fuzzy regulátoru nastaveny na následující hodnoty:

	min.	max.
E	-5	5
ΔE	-10	10
ΔU	-100	100

S těmito parametry se podařilo zrychlit odezvy na náhlý pokles tlaku, takže fuzzy regulátor reagoval dříve a byl schopen vyrovnat tlak na požadovanou hodnotu. Průběh tlaku na transformátoru DIT s hydraulickým utahovákem je zobrazen v grafu na obrázku 5.7.

5.6 Nasazení transformátoru v praxi

Aplikace výsledků řešení umožní realizovat mnohé projekty tak, že tlakový zdroj pracující na bázi minerálního oleje bude oddělen od prostředí s nebezpečím požáru, výbuchu nebo hygienicky nezávadného prostředí a v daném prostředí bude nahrazen hydromotorem na vodu nebo vodní emulzi. Inverzní zařízení pak bude sloužit tam, kde je naopak k dispozici vodní tlakový zdroj a hydraulické prvky na minerální olej jsou v běžném klasickém prostředí, takže je není nutné nahrazovat drahými prvky na vodu.



Obrázek 5.7: Průběh tlaku na transformátoru DIT s hydraulickým utahovákem.

Díky vyvinutému zařízení je možné použít hydraulické pohony i v oblastech, které se vyznačují nebezpečím požáru, výbuchu nebo do oblastí potravinářského průmyslu, lékařství, chemie, kde existují přísná hygienická opatření a pohony s minerálními oleji jsou prakticky nepoužitelné.

V průběhu roku 2005 již došlo k praktické aplikaci neinverzního pístového transformátoru při realizaci testovacího zařízení na výzkum jevů souvisejících se vstřikováním paliva do dieselových spalovacích motorů s tlaky 150 bar na primární straně a 1500 bar na sekundární straně. Jedná se vlastně o jednosměrný transformátor s multiplikací pracovního tlaku při přenosu tlakové energie směrem minerální olej → nafta. V průběhu měsíců 9/2005 a 10/2005 bylo zkušební zařízení úspěšně ověřeno, přičemž projektovaných parametrů bylo dosaženo v plném rozsahu.

5.7 Shrnutí

Celkově je možno řešení projektu ze strany Ostravské Univerzity shrnout do následujících několika bodů:

1. Byla vytvořena aplikace realizující fuzzy regulaci. Jádro aplikace představující vlastní fuzzy regulátor používá COM objekt z programového balíku LFLC 2000. Tato aplikace byla vyvíjena a upravována v celém průběhu řešení projektu podle aktuálních potřeb. Pomocí této aplikace bylo umožněno provádět a zaznamenávat měření a byla klíčovým nástrojem k ladění jazykového popisu a k určování kontextů veličin fuzzy regulátoru. Nejprve byla navržena pro regulaci simulátoru procesu transformátoru a v závěrečné fázi projektu, kdy již bylo možno provádět měření na prototypech transformátorů byla aplikace přizpůsobena na řízení těchto prototypů.

2. Byl navržen a odladěn jazykový popis fuzzy PI regulátoru odpovídající danému procesu. Při

jeho návrhu se vycházelo z univerzálního jazykového popisu pro PI regulaci a jeho modifikací a postupným laděním se dospělo ke konečnému tvaru, který popisuje současné chování fuzzy regulátoru. Jazykový popis fuzzy regulátoru byl v raném stádiu navržen pro výše zmíněný simulátor procesu transformátoru tlakové energie a dále jej již nebylo zapotřebí měnit. K návrhu a ladění jazykového popisu byl použit program LFLC 2000.

3. Byly určeny hodnoty parametrů kontextů veličin fuzzy regulátoru z provedených měření, čímž byl dokončen návrh fuzzy regulátoru jako celku. Obdobně jako v předchozích bodech byly nejprve tyto hodnoty určeny pro simulátor transformátoru, čímž se dosáhlo dílčího úspěchu v prokázání toho, že fuzzy regulace bude v tomto případě uspokojivě použitelná. V závěrečné fázi projektu bylo možno určit hodnoty kontextů veličin regulátoru z uskutečněných testovacích měření na prototypech transformátorů tlakové energie.

4. Fuzzy regulátor byl formou tabulky hodnot řídící funkce připraven pro implementaci do jednočipu.

E	&	ΔE	→	ΔU
-bi	&	-bi	→	-bi
-bi	&	-me	→	-bi
-bi	&	-sm	→	-bi
-bi	&	sm	→	-me
-bi	&	me	→	ze
-bi	&	bi	→	ze
-bi	&	ze	→	-me
-me	&	ze	→	-ro sm
-me	&	bi	→	ve sm
-me	&	me	→	si sm
-me	&	sm	→	-ex sm
-me	&	-sm	→	-me
-me	&	-me	→	-bi
-me	&	-bi	→	-bi
-sm	&	ze	→	-ve sm
-sm	&	bi	→	sm
-sm	&	me	→	si sm
-sm	&	sm	→	-ex sm
-sm	&	-sm	→	-sm
-sm	&	-me	→	-me
-sm	&	-bi	→	-bi
-ve sm	&	sm	→	ex sm
-ve sm	&	ve sm	→	ze
-ve sm	&	-sm	→	-si sm
-ve sm	&	-me	→	-ve sm

E	&	ΔE	→	ΔU
ve sm	&	-sm	→	-si sm
ve sm	&	-ve sm	→	ze
ve sm	&	sm	→	si sm
ve sm	&	me	→	ve sm
sm	&	-bi	→	-ml sm
sm	&	-me	→	-sm
sm	&	-sm	→	ex sm
sm	&	sm	→	sm
sm	&	me	→	me
sm	&	bi	→	bi
sm	&	ze	→	ve sm
me	&	-bi	→	-ve sm
me	&	-me	→	-ve sm
me	&	-sm	→	ex sm
me	&	sm	→	me
me	&	me	→	bi
me	&	bi	→	bi
me	&	ze	→	ro sm
bi	&	ze	→	me
bi	&	bi	→	bi
bi	&	me	→	bi
bi	&	sm	→	bi
bi	&	-sm	→	me
bi	&	-me	→	ze
bi	&	-bi	→	ze
ze	&	ze	→	ze

Tabulka 5.1: Báze pravidel fuzzy regulátoru.

ΔU		ΔE								
		-bi	-me	-sm	-ve sm	ze	ve sm	sm	me	bi
E	-bi	-bi	-bi	-bi		-me		-me	ze	ze
	-me	-bi	-bi	-me		-ro sm		-ex sm	si sm	ve sm
	-sm	-bi	-me	-sm		-ve sm		-ex sm	si sm	ve sm
	-ve sm		-ve sm	-si sm			ze	ex sm		
	ze					ze				
	ve sm			-si sm	ze			si sm	ve sm	
	sm	-ml sm	-sm	ex sm		ve sm		sm	me	bi
	me	-ve sm	-ve sm	ex sm		ro sm		me	bi	bi
	bi	ze	ze	me		me		bi	bi	bi

Tabulka 5.2: Dvojrozměrná verze báze pravidel fuzzy regulátoru.

Kapitola 6

Závěr

V této práci jsme se zaměřili na zkoumání některých vlastností fuzzy transformace souvisejících především s její algoritmickou implementací, výpočetní složitostí a její aplikovatelností na filtraci signálu. V oblasti nástrojů pro podporu fuzzy modelování byla vytvořena varianta fuzzy Petriho sítě umožňující vizualizaci a návrh hierarchických fuzzy IF–THEN pravidel. Pro fuzzy regulaci a řízení byl navržen adaptivní fuzzy regulátor nastavující kontexty výkonného fuzzy regulátoru pro dosažení vyšší přesnosti regulace. Podrobnější seznam výsledků této práce je uveden v úvodní části 1.3.

Během řešení výše uvedených úloh vyvstaly některé nové problémy, které budou předmětem dalšího zkoumání a vývoje.

V současné době již došlo k výraznému posunu ve vývoji aplikace na analýzu a předpověď časových řad. Další vývoj bude zaměřen jednak na vytvoření uživatelsky přívětivého rozhraní, ale především bude spočívat v rozšíření variability použitých algoritmů. Například se jedná o zahrnutí možnosti multiplikativního charakteru sezónní složky vedle stávajícího aditivního modelu nebo předpověď trendu na základě logické dedukce vycházející z kombinace několika předchozích hodnot a jejich zpětných diferencí a další.

Pro kompresi obrázků založené na dvojrozměrné F-transformaci bude snaha o zlepšení kompresního poměru směřována na členění obrázku na oblasti s malými změnami barev přičemž v úvahu bude brána také triangulární F-transformace, která má výhodu v lokálním charakteru změn způsobených přidáním nového uzlu do fuzzy rozkladu.

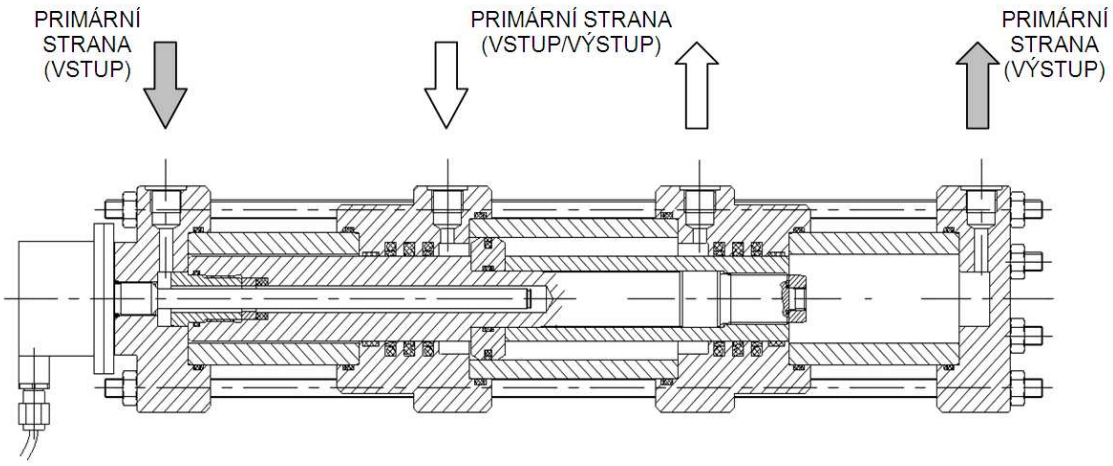
U triangulární fuzzy transformace je otevřený problém nalezení vhodné datové struktury pro reprezentaci fuzzy rozkladu s ohledem na lokalizaci spádové oblasti příslušející bodu z definičního oboru aproximované funkce.

V souvislosti s filtrací signálu pomocí fuzzy filtru vznikají otázky související se statistickým rozdělením šumové složky signálu, kdy nás především zajímá vliv filtru na rozptyl a případně i některé další charakteristiky výstupního signálu.

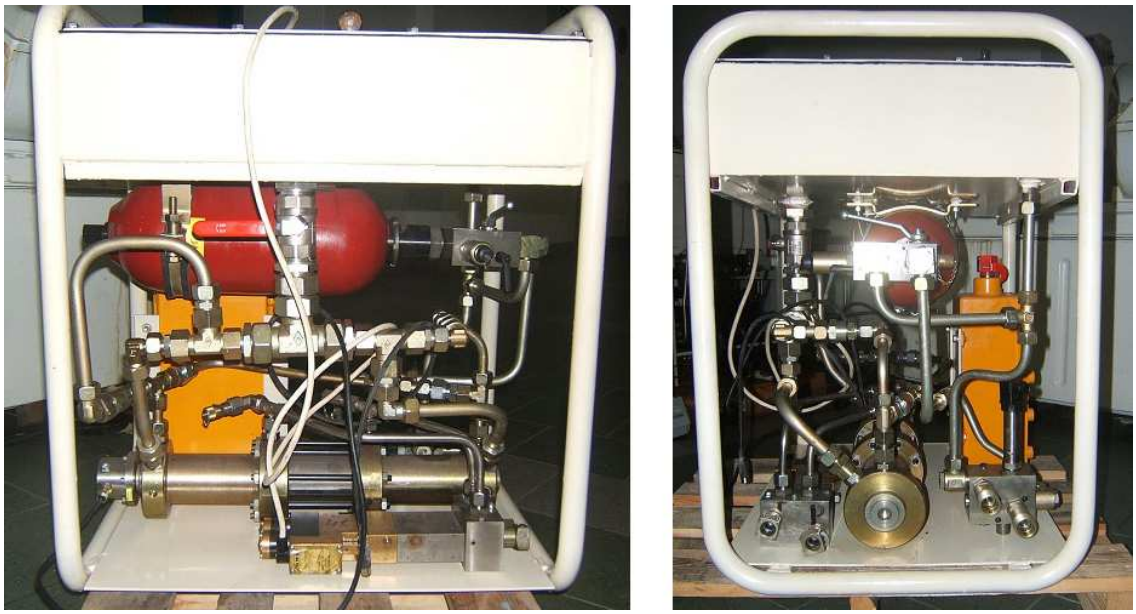
Pro modelování složitějších fuzzy systémů, včetně návrhu hierarchických a hybridních fuzzy regulátorů, se jeví jako vhodná varianta kombinace některého druhu klasické Petriho sítě s fuzzy Petriho sítěmi, definovanými v této práci. Petriho sítě totiž samy o sobě poskytují výkonné prostředky pro modelování kauzality, nedeterminismu a paralelismu, což ve spojení s možností modelování vágnosti přirozeného jazyka pomocí fuzzy logiky z nich dělá ještě silnější nástroj pro modelování komplexních systémů. Z toho důvodu je jejich další zkoumání hlavním předmětem pokračování této práce do budoucna.

Příloha A

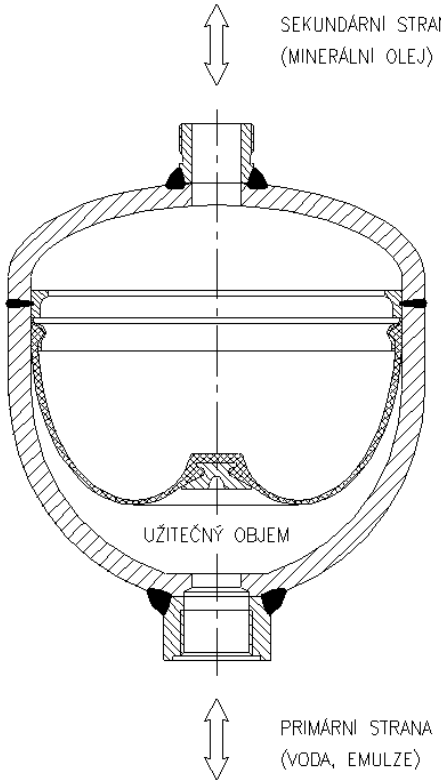
Transformátor tlakové energie – obrazová příloha



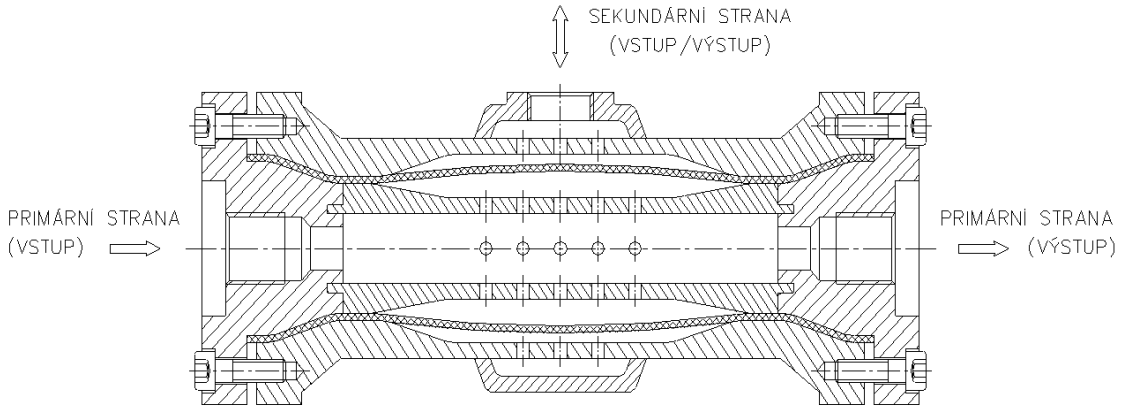
Obrázek A.1: Nákres dvojčinného inverzního transformátoru (DIT).



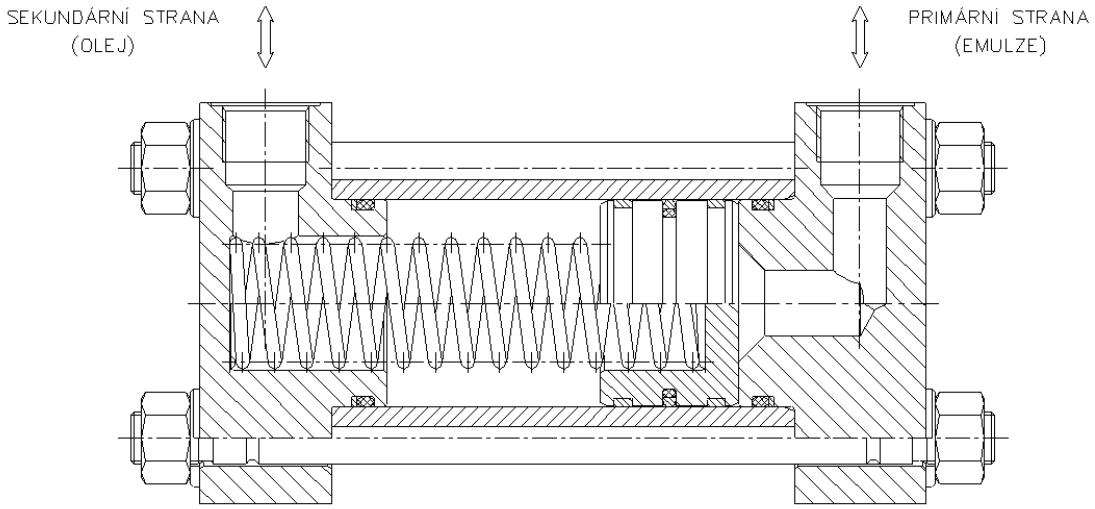
Obrázek A.2: Vývojový funkční prototyp transformátoru DIT.



Obrázek A.3: Řez membránovým inverzním transformátorem (MIT).

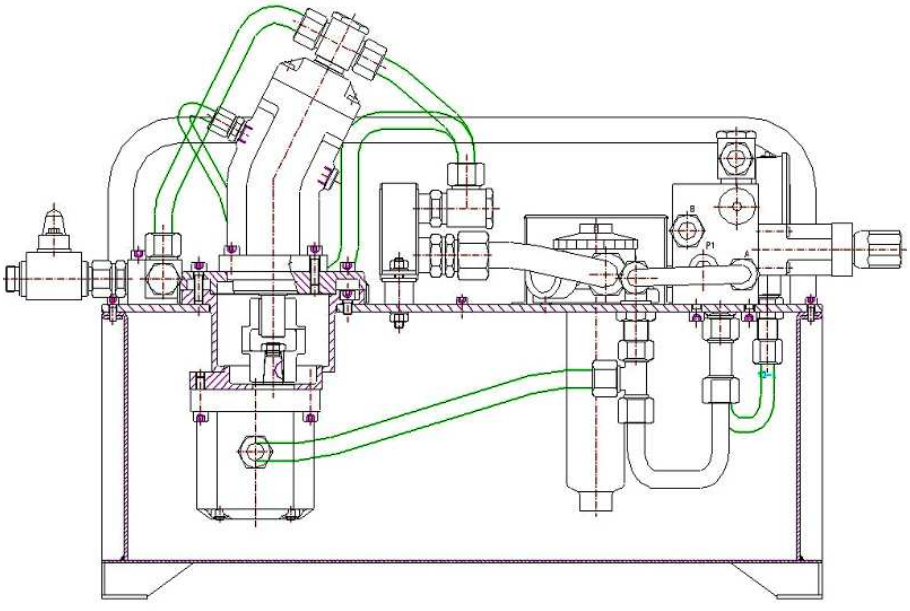


Obrázek A.4: Nákres pryžového inverzního transformátoru (QIT).



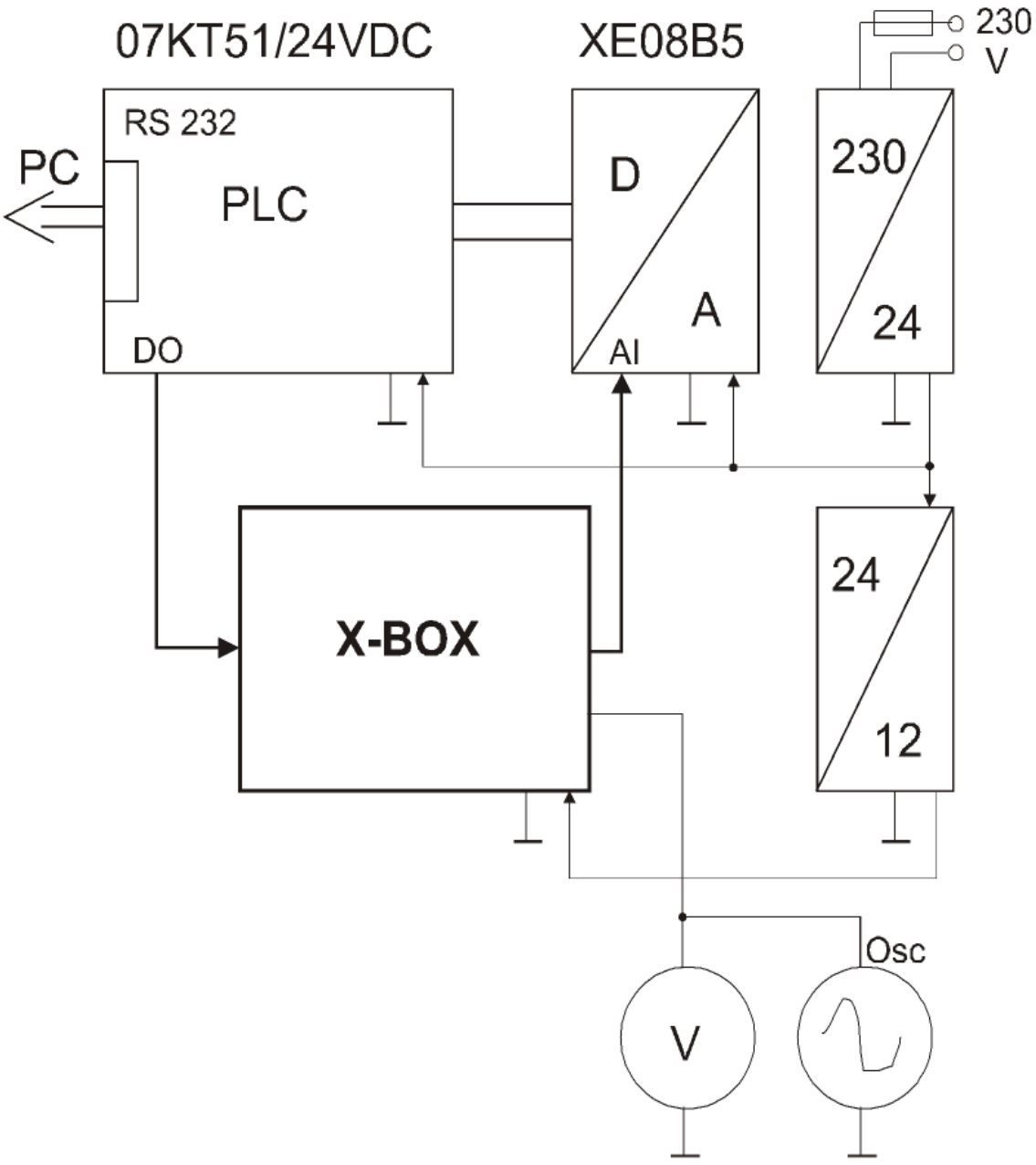
Obrázek A.5: Řez pístovým inverzním transformátorem (PIT).

Obrázek A.6: Vývojové funkční prototypy transformátorů (zleva MIT, PIT, QIT).

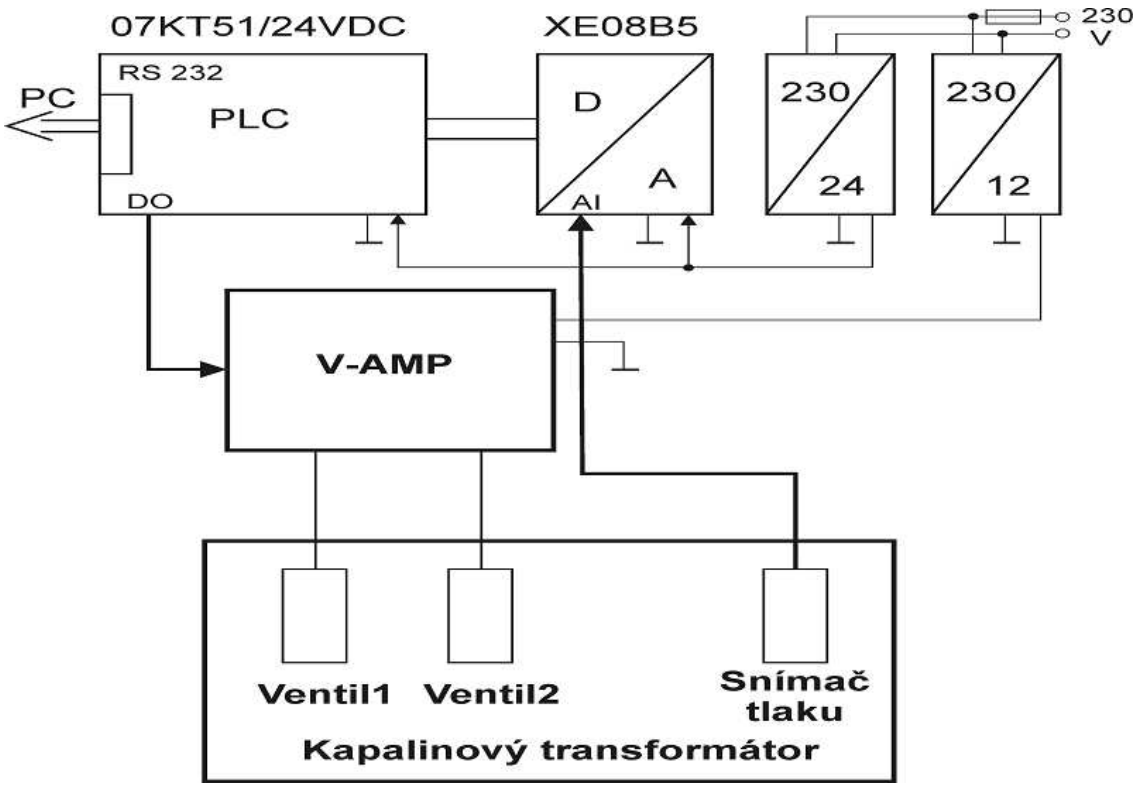


Obrázek A.7: Řez transformátorem RIT.

Obrázek A.8: Vývojový funkční prototyp transformátoru RIT.



Obrázek A.9: Elektronické schéma zapojení fuzzy regulátoru se simulátorem děje X-BOX.



Obrázek A.10: Elektronické schéma zapojení ovládání funkčního prototypu transformátoru.

Literatura

Vlastní publikace

[1] Dvořák, A., Habiballa, H., Novák, V., Pavliska, V. (2003), The concept of LFLC 2000 – its specificity, realization and power of applications. *Computers in Industry*, sv. **51**, 269–280.

[2] Dvořák, A., Habiballa, H., Pavliska, V., et al. (2003), LFLC 2000 + MATLAB/SIMULINK – Systém pro universální aplikace fuzzy logiky, *Sborník přednášek ze 7. ročníku konference Inteligentní systémy v praxi*, Ostrava: AD&M, 47–48.

[3] Habiballa, H., Novák, V., Dvořák, A., Pavliska, V. (2003), Využití softwarového balíku LFLC 2000, *2nd International Conference Aplimat 2003*, Bratislava: Slovenská technická univerzita, 355–358.

[4] Dvořák, A., Novák, V., Pavliska, V. (2006), Rules, Inferences and Robust Approximation at Work. *ERCIM News*, sv. **64**, 31–32.

[5] Štěpnička, M., Lughofer, E., Pavliska, V. (2006), Comparison of Data-Driven Fuzzy Modelling Methods tested on NOx Data, *Abstracts of the FLLL/SCCH Master and PhD Seminar*, Hagenberg/Linz: FLLL/SCCH, 33-39.

[6] Blažek, J., Habiballa, H., Pavliska, V. (2005), Vybrané heuristiky pro globální optimalizaci a jejich implementace v MATLABu, *Proceedings of the International Conference Technical Computing Prague 2005*. Praha: VŠCHT, **19**.

[7] Habiballa, H., Pavliska, V., Bražina, D. (2005), Objektová knihovna evolučních algoritmů, *Sborník konference Tvorba softwaru 2005*, Ostrava: Tanger, 18–24.

[8] Tvrdík, J., Pavliska, V., Habiballa, H. (2007), Matlab Program Library for Box-Costrained Global Optimization, *Proceedings of International Conference Aplimat 2007*, Bratislava: Slovenská technická univerzita, 463–470.

[9] Huňka, F., Pavliska, V. (2008), Object Oriented Approach in Optimization of Fuzzy Transform, In: *New Aspects of Computers, Proceedings of the WSEAS International Conference on COMPUTERS*, Heraklion, Greece, 1066–1071.

[10] Knybel, J., Pavliska, V. (2005), Representation of Fuzzy IF-THEN rules by Petri Nets, In: *ASIS 2005*, Přerov, Ostrava, MARQ, 121–125.

[11] Perfilieva, I., Novák, V., Pavliska, V., et al. (2007), Prediction of Time Series by Soft Computing Methods, *Proc. 10th Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty*, Praha: UTIA AV ČR, 119–129.

[12] Novák, V., Štěpnička, M., Perfilieva, I., Pavliska, V. (2008), Analysis of periodical time series using soft computing methods, *Computational Intelligence in Decision and Control*, Singapore: World Scientific, 55–60.

[13] Perfilieva, I., Novák, V., Pavliska, V., et al. (2008), Analysis and Prediction of Time Series Using Fuzzy Transform, *WCCI 2008 Proceedings*, Hong Kong: IEEE Computational Intelligence Society, 3875–3879.

[14] Perfilieva, I., Pavliska, V., Vajgl, M., et al. (2008), Advanced Image Compression on the Basis of Fuzzy Transforms, *IPMU 08 Proceedings*, Malaga: University of Malaga, 1167–1174.

[15] Górecki, I., Mrůzek, P., Sobotíková, M., et al. (2005), Výzkum a vývoj zařízení pro transformaci přenosu tlakové energie na bázi minerální olej – voda a realizace technologie řízení tohoto procesu, Oponovaná závěrečná zpráva o řešení projektu FD-K3/033 (2003–2005), Ostrava.

Použitá literatura

[16] Novák V. (1986 a 1990), *Fuzzy množiny a jejich aplikace*. Státní nakladatelství technické literatury, Praha.

[17] Novák V. (2000), *Základy fuzzy modelování*. Technická literatura BEN, Praha.

[18] Novák, V., Lehmke, S.(2006), Logical structure of fuzzy IF–THEN rules. *Fuzzy Sets and Systems*, **157**, 2003-2029.

[19] Novák, V., Perfilieva, I., Močkoř, J.(1999), *Mathematical Principles of Fuzzy Logic.* Kluwer, Boston, Dordrecht, London.

[20] Novák, V., Perfilieva, I (1999), Evaluating Linguistic Expressions and Functional Fuzzy Theories in Fuzzy Logic. In: L. A. Zadeh a J. Kacpryk (eds.): *Computing with Words in Information/Intelligent Systems 1*, Springer-Verlag, Heidelberg, 383–406.

[21] Novák, V., (2005), Perception-Based Logical Deduction. In: Reusch, B. (Ed.): *Computational Intelligence, Theory and Applications*, Springer, Heidelberg, 241–253.

[22] Novák, V., Perfilieva, I. (2004), On the Semantics of Perception-Based Fuzzy Logic Deduction, *Int. Journal of Intelligent Systems*, **19**, 1007–1031.

[23] Perfilieva, I., Chaldeeva, E.(2001), Fuzzy transformation and its applications. In: *4th Czech – Japan Seminar on Data Analysis and Decision Making under Uncertainity*, 116–124.

[24] Perfilieva, I.(2004), Fuzzy transforms. In: *Transactions on Rough Sets II. Rough Sets and Fuzzy Sets, J.F. Peters and A. Skowron (Eds.), LNCS 3135*, 63–81.

[25] Perfilieva, I.(2006), Fuzzy transforms: Theory and Applications. *Fuzzy Sets and Systems*, **157**, 993–1023.

[26] Perfilieva, I.(2007), Fuzzy Transforms: A Challenge to Conventional Transforms, In: *Advances in Imaging and Electron Physics. vol. 147*, 137–196.

[27] Štěpnička, M., Valášek, R.(2003), Fuzzy transforms for function with two variables. In: *Proc. of the 6th Czech – Japan Seminar on Data Analysis and Decision Making under Uncertainity*, 100–107.

[28] Zadeh, L. A. (1973), *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*, Trans. IEEE, SMC 3.

[29] Kickert, W., J., M., Mamdani, F., H. (1978), Analysis of a Fuzzy Logic Controller, *Fuzzy Sets and Systems*, **1**.

[30] Jantzen, J. (2007), *Foundations of Fuzzy Control*, John Witley and Sons, Inc., England.

[31] Pokorný, M. (1995), *Řídící systémy se znalostí bází*, Vysoká škola báňská, Ostrava.

[32] Jura, P. (2003), *Základy fuzzy logiky pro řízení a modelování*, Vutium, Brno.

[33] Vysoký, P. (1996), *Fuzzy řízení*, České vysoké učení technické, Praha.

[34] Čéška, M. (1994), *Petriho sítě*, Akademické nakladatelství CERM, Brno.

[35] Janoušek, V. (1998), *Modelování objektů Petriho sítěmi*, Disertační práce, Ústav informatiky a výpočetní techniky FEI VUT, Brno.

[36] Billington, J., et al. (2003), The Petri Net Markup Language: Concepts, Technology, and Tools, In: *conference proceedings of the ICATPN 2003 to be held in Eindhoven*, Netherlands.

[37] Weber, M., Kindler, E. (2002), The Petri Net Markup Language, In: *Petri Net Technology for Communication Based Systems* within the LNCS series: *Advances in Petri Nets.*

[38] Hauptmann, J., et al. (1998), *The Petri Net Kernel - Documentation of the Application Interface*, Humboldt University, Berlin.

[39] Harold, E., R. (2001), *XML Bible, Second Edition*, Hungry Minds, Inc., New York.

[40] Štěpnička, M., Polakovič, O. (2008) A neural network approach to the fuzzy transform, *Fuzzy Sets and Systems.*

[41] Price, K., V., Storm, R., M., Lampinen, J., A. (2005), *Differential Evolution,* Springer Verlag, Berlin–Heidelberg.

[42] Tvrdík, J. (2007), Adaptive Differential Evolution: Application to Nonlinear Regression, In *Proceedings of the International Multiconference on Computer Science and Information Technology,* Vol. 1, No. 2, 193–202.

[43] Tvrdík, J., Křivý, I. (2007), Competitive Self-Adaptation in Evolutionary Algorithms, In *5th Conference of European Society for Fuzzy Logic and Technology*, Ostrava, University of Ostrava, 251–258.

[44] Perfilieva, I., Valášek, R. (2005), Fuzzy Transforms in Removing Noise, In Reusch, B. (Ed.), *Computational Intelligence, Theory an Applications*, Springer, Heidelberg, 225–234.

[45] Perfilieva, I. (2006), *Fuzzy Transforms and Their Applications to Image Compression*, 3849. vyd. Heidelberg: Springer, 19–31.

[46] Di Martino, F., Loia, V., Perfilieva, I., et al. (2008), An image coding/decoding method based on direct and inverse fuzzy transforms, *International Journal of Appr. reasoning*, sv. **48**, 110–131.

[47] Zadeh, L. A. (1975), The concept of a linguistic variable and its application to approximate reasoning I, II, III, Inf. Sci. 199–257, 301–357, 43–80.

[48] Perfilieva, I. (2003), Fuzzy approach to solution of differential equations with imprecise data: Application to Reef Growth Problem, In: *R.V. Demicco, G. J. Klir (Eds.) Fuzzy Logic in Geology*, Academic Press, Amsterdam, 275–300.

[49] Štěpnička, M., Valášek, R. (2005), Numerical Solution of Partial Differential Equations with Help of Fuzzy Transform, *Proceedings of the FUZZ-IEEE 2005*, Reno, Nevada: Fuzz-IEEE 2005, 1104–1109.

[50] Štěpnička, M., Valášek, R. (2004) Fuzzy Transforms and Their Application to Wave Equation, *Journal of Electrical Engineering*, roč. 55, sv. **12**, 7–10.

Rejstřík